HỌC VIỆN KỸ THUẬT MẬT MÃ

KHOA AN TOÀN THÔNG TIN

BÁO CÁO MÔN HỌC

CƠ SỞ AN TOÀN THÔNG TIN

Đề tài:

TÌM HIỂU VỀ GIAO THỨC HTTP VÀ HTTPS

Hà Nội, 10-2023

HỌC VIỆN KỸ THUẬT MẬT MÃ
KHOA AN TOÀN THÔNG TIN

BÁO CÁO MÔN HỌC
CƠ SỞ AN TOÀN THÔNG TIN

Đề tài:
TÌM HIỂU VỀ GIAO THỨC HTTP VÀ HTTPS

Sinh viên thực hiện: TẠ XUÂN CƯỜNG – AT170107
ĐỖ CÔNG MINH – AT170634
CAO ĐẮC QUÂN – AT170640

Nhóm 21

Hà Nội, 10-2023

MỤC LỤC

DANH MỤC VIẾT TẮT

STT	Từ viết tắt	Tên đầy đủ
1	TCP/IP	Transmission Control Protocol/Internet Protocol
2	HTTP	Hypertext Transfer Protocol
3	HTTPS	Hypertext Transfer Protocol Secure
4	HTML	Hypertext Markup Language
5	URL	Uniform Resource Locator
6	DNS	Domain Name System
7	VPN	Virtual Private Network
8	API	Application Programming Interface
9	SSL	Secure Sockets Layer
10	TLS	Transport Layer Security
11	CA	Certificate Authority

DANH MỤC HÌNH ẢNH

TÓM TẮT ĐỀ TÀI

- Tổng quan về giao thức HTTP, cách thức hoạt dộng giữa máy khác và máy chủ, các thành phần trong giao thức HTTP.
- Tổng quan về giao thức HTTPS, cách thức hoạt động của HTTPS, các thành phần trong gói tin HTTPS. So sánh HTTP với HTTPS.
- Tiến hành thực nghiệm sử dụng công cụ Wireshark để phân tích gói tin HTTP và HTTPS.

LỜI NÓI ĐẦU

Trong thời đại của cuộc cách mạng số hóa, giao thức HTTP (Hypertext Transfer Protocol) và HTTPS (Hypertext Transfer Protocol Secure) đóng vai trò quan trọng cơ sở hạ tầng internet. Chúng không chỉ là những khái niệm kỹ thuật mà còn là nền tảng của mọi trải nghiệm trực tuyến của chúng ta, từ việc duyệt web hàng ngày, truy cập ứng dụng trực tuyến, đến gửi và nhận dữ liệu qua mạng. Những giao thức này là những người bạn đồng hành vô cùng đáng tin cậy, luôn ẩn sau màn hình máy tính và điều hành như "giao tiếp ngôn ngữ" của internet.

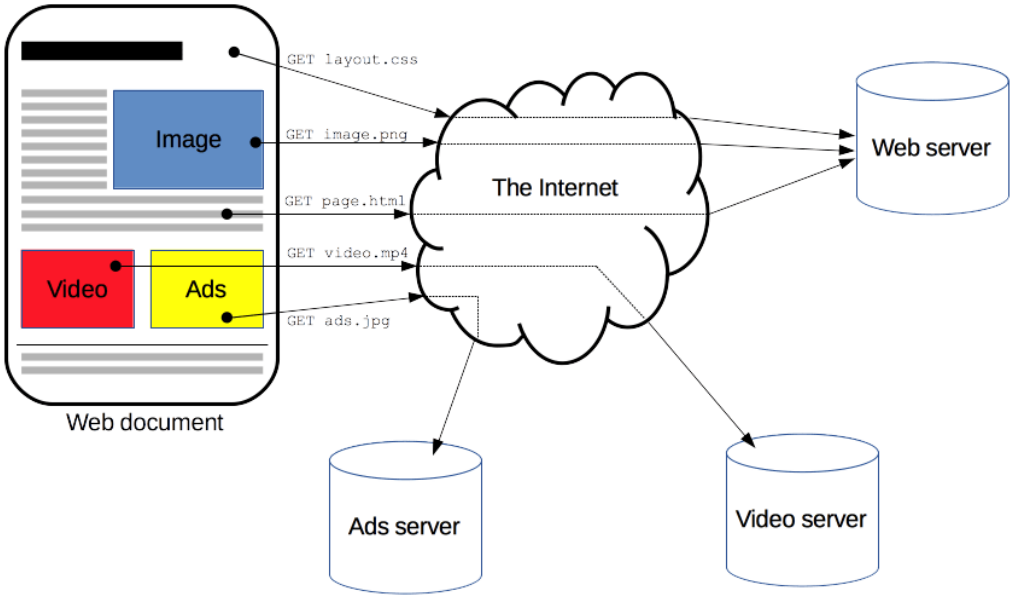
Để có thể vận dụng tốt được các chức năng của hai giao thức HTTP và HTTPS cũng như có thể tự bảo vệ dữ liệu, thông tin của mình trên mạng Internet. Chúng ta cần hiểu rõ được các chức năng cũng như ưu nhược điểm của chúng.

Báo cáo này tìm hiểu về cách hoạt động, tầm quan trọng, và tác động của giao thức HTTP và HTTPS trong thế giới mạng hiện đại. Trong quá trình nghiên cứu, chúng tôi đã đặt ra các câu hỏi quan trọng về bảo mật, hiệu suất, và tương lai phát triển của hai giao thức này.

CHƯƠNG 1: TỔNG QUAN VỀ GIAO THỨC HTTP

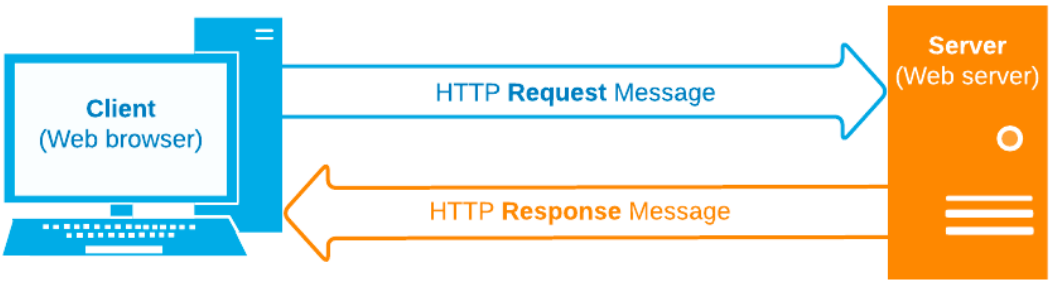
1.1 Giao thức HTTP

HTTP (Hypertext Transfer Protocol) là một giao thức nằm ở tầng ứng dụng của bộ giao thức TCP/IP, được dùng trong quá trình giao tiếp giữa Client và Web Server.



Hình 1. 1: Tổng quan về giao thức HTTP

HTTP hoạt động dựa trên mô hình Client – Server. Khi client (thường là browser) gửi một gói tin yêu cầu (request packet) tới web server, web server sẽ tiếp nhận, xử lý yêu cầu và trả về một gói tin phản hồi (response packet). Sau đó browser sẽ hiển thị gói phản hồi này dưới dạng văn bản, hình ảnh, âm thanh,… mà người dùng có thể hiểu được.

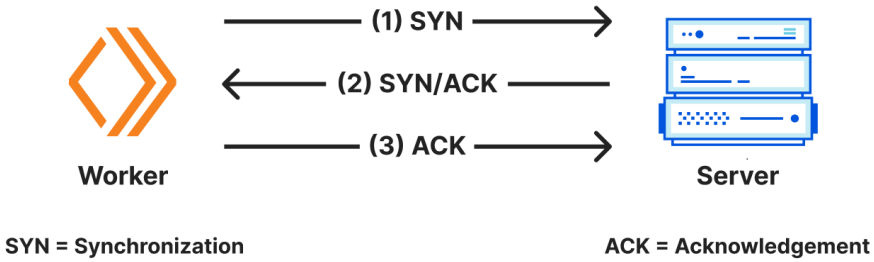


Hình 1. 2: Mô hình hoạt động của HTTP

Giao thức HTTP hỗ trợ nhiều phương thức khác nhau như GET, POST, PUT, DELETE,… Mỗi phương thức có mục đích và chức năng riêng cho phép thực hiện các hoạt động khác nhau đối với máy chủ web.

1.2 Cách hoạt động của HTTP

1.2.1 Quá trình bắt tay ba bước



Hình 1. 3: Giao thức bắt tay ba bước

Trước khi client và server có thể giao tiếp với nhau bằng giao thức HTTP, chúng phải thiết lập một kết nối TCP. Quá trình này được gọi là quá trình bắt tay ba bước.

Bước 1: Client gửi một gói tin TCP có cờ SYN đến server. Gói tin này chứa một số ngẫu nhiên được gọi là Sequence Number, để định dạng gói tin và xác định vị trí của dữ liệu trong luồng truyền.

Bước 2: Server nhận được gói tin TCP chứa cờ SYN, sau đó gửi lại một gói tin TCP chứa cờ SYN và cờ ACK để trả lời. Gói tin này cũng chứa một số ngẫu nhiên, được gọi là Acknowledgment Number, và một phản hồi cho Sequence Number đã nhận từ client ở bước 1.

Bước 3: Client gửi gói tin TCP cuối cùng chứa cờ ACK cho server để xác nhận rằng nó đã nhận được gói SYN-ACK và kết nối đã được thiết lập.

Quá trình bắt tay ba bước đảm bảo rằng cả hai máy tính hiểu về việc thiết lập kết nối và đã chuẩn bị sẵn sàng để truyền dữ liệu. Nó cũng cung cấp tính năng kiểm soát và xác thực trong quá trình thiết lập kết nối mạng, đóng vai trò quan trọng trong đảm bảo tính an toàn và đáng tin cậy của dữ liệu truyền qua mạng.

1.2.2 Quá trình giao tiếp bằng HTTP

Sau khi hai máy tiến hành mở kết nối bằng quá trình bắt thay ba bước. Chúng sẽ bắt đầu việc giao tiếp với nhau bằng quá trình gửi và nhận gói tin HTTP, việc trao đổi đó gồm các bước sau đây:

Bước 1: Client gửi yêu cầu HTTP

- Trình duyệt hoặc ứng dụng tạo một yêu cầu HTTP dưới dạng gói tin HTTP request.
- Gói tin request bao gồm các phần sau:
    - Phương thức HTTP (ví dụ: GET, POST, PUT, DELETE): Xác định cách trình duyệt muốn tương tác với máy chủ.
    - URL: Địa chỉ của trang web hoặc tài nguyên cụ thể.

    - Header HTTP: Các thông tin bổ sung như tiêu đề yêu cầu, các thông số và cookie.
    - Body (nếu cần): Dữ liệu bổ sung, chẳng hạn như dữ liệu biểu mẫu nếu phương thức là POST.
- Gói tin request được chuyển đi từ trình duyệt hoặc ứng dụng của người dùng đến máy chủ web qua kết nối TCP/IP đã thiết lập.

Bước 2: Máy chủ web xử lý yêu cầu

- Máy chủ web nhận gói tin request HTTP từ kết nối TCP/IP.
- Máy chủ web xử lý yêu cầu bằng cách:
    - Xác định tài nguyên hoặc thông tin cần truy cập dựa trên URL và phương thức.
    - Thực hiện xử lý logic nếu cần thiết, chẳng hạn như kiểm tra quyền truy cập, thực hiện các truy vấn cơ sở dữ liệu, xử lý dữ liệu, v.v.

Bước 3: Máy chử gửi phản hồi HTTP

- Máy chủ web tạo một gói tin HTTP response để trả lời yêu cầu.
- Gói tin response bao gồm các phần sau:
    - Mã trạng thái HTTP (ví dụ: 200 OK nếu thành công hoặc mã lỗi khác nếu có lỗi).
    - Header HTTP: Các thông tin bổ sung như loại nội dung, kích thước tài liệu, thời gian hết hạn, v.v.
    - Body: Nội dung thực sự của tài liệu hoặc dữ liệu, chẳng hạn như trang web, hình ảnh, văn bản, hoặc dữ liệu JSON/XML.

Bước 4: Client nhận phản hồi và hiển thị kết quả

Gói tin response được gửi từ máy chủ web đến trình duyệt hoặc ứng dụng của người dùng qua kết nối TCP/IP.

- Trình duyệt hoặc ứng dụng nhận gói tin response và xử lý nó.
- Trình duyệt hiển thị nội dung hoặc tài liệu từ gói tin response trên màn hình của người dùng, cho phép họ tương tác với nó.

Đây là quy trình chi tiết về cách gửi và nhận gói tin HTTP trong giao thức HTTP, cho phép người dùng truy cập và tương tác với các trang web và dịch vụ trực tuyến trên Internet.

1.3 Thiết lập kết nối

Khi client muốn kết nối với server bằng giao thức TCP (Transmission Control Protocol), quá trình này diễn ra theo các bước cơ bản như sau:

Bước 1: Xác định IP và cổng server: Trước khi client có thể kết nối với server, nó cần biết địa chỉ IP của server và cổng mạng mà server đang lắng nghe. Địa chỉ IP xác định máy chủ cụ thể và cổng xác định dịch vụ cụ thể trên máy chủ.

Bước 2: Thiết lập kết nối TCP/IP: Client sử dụng thư viện hoặc API TCP/IP có sẵn trong ngôn ngữ lập trình của mình để thiết lập một kết nối TCP với server. Quá trình này bao gồm các bước sau:

- Tạo socket: Client tạo một socket (địa chỉ IP và cổng) để thiết lập kết nối.
- Xác định địa chỉ IP và cổng của server: Client cung cấp địa chỉ IP và cổng của server mà nó muốn kết nối đến thông qua socket.
- Gửi yêu cầu kết nối (TCP handshake): Client gửi một yêu cầu kết nối đến server sử dụng gói tin TCP handshake. Gói tin này bao gồm thông tin về socket của client.

Bước 3: Server chấp nhận kết nối: Khi server nhận được yêu cầu kết nối từ client, nó kiểm tra xem có thể chấp nhận kết nối này hay không. Nếu server chấp nhận, quá trình tiếp tục.

Bước 4: Thiết lập kết nối hoàn chỉnh: Sau khi server chấp nhận kết nối, nó gửi lại một gói tin TCP handshake để xác nhận việc thiết lập kết nối. Sau khi client nhận được gói tin này, kết nối TCP đã được thiết lập hoàn chỉnh và client và server có thể trao đổi dữ liệu qua kết nối này.

Bước 5: Truyền dữ liệu: Client và server có thể bắt đầu truyền dữ liệu qua kết nối TCP. Dữ liệu được chia thành các gói tin TCP nhỏ và được đóng gói và giải gói tại cả hai đầu để đảm bảo tính toàn vẹn và độ tin cậy của dữ liệu.

Bước 6: Đóng kết nối: Khi client hoàn thành truyền dữ liệu hoặc không còn cần kết nối nữa, nó có thể gửi yêu cầu đóng kết nối đến server. Server cũng có thể gửi yêu cầu đóng kết nối nếu nó hoàn thành việc truyền dữ liệu hoặc xảy ra lỗi. Sau khi cả hai bên đều đồng ý đóng kết nối, nó sẽ được đóng lại và không thể sử dụng nữa.

Đó là cách mà client kết nối với server bằng giao thức TCP. Quá trình này đảm bảo tính toàn vẹn, độ tin cậy và sự đồng bộ trong việc truyền tải dữ liệu giữa client và server.

1.4 Cấu trúc gói tin HTTP

Một gói tin HTTP có chung cấu trúc bao gồm 4 phần chính là start line, header, empty line và body.

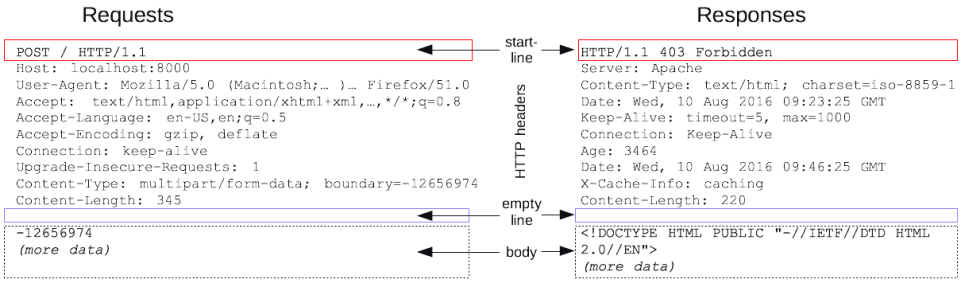
Start line: mô tả các yêu cầu sẽ được thực hiện hoặc trạng thái thành công hay thất bại. Dòng bắt đầu này luôn là một dòng duy nhất.

HTTP header: gồm một bộ tiêu đề HTTP tùy chọn chỉ định yêu cầu hoặc mô tả nội dung có trong thư. Trong đó có 4 loại header theo ngữ cảnh:

- General Header: Loại header này được áp dụng trên cả requese header và response header nhưng không ảnh hưởng đến nội dung của phần body.
- Request Header: Loại header này chứa thông tin về yêu cầu của client.
- Response Header: Loại header này chứa vị trí của nguồn đã được client yêu cầu.
- Entity Header: Loại header này chứa thông tin về phần body như loại MIME hay độ dài nội dung.

Empty line: cho biết tất cả thông tin meta cho yêu cầu đã được gửi.

Body: chứa yêu cầu của client hoặc dữ liệu phản hồi của server (như nội dung của biểu mẫu HTML) hoặc tài liệu được đính kèm với phản hồi. Sự hiện diện của phần body và kích thước của nó được chỉ định bởi start line và HTTP header.



Hình 1. 4: Cấu trúc gói tin HTTP

1.4.1 Cấu trúc gói tin HTTP Request

1.4.1.1 Start line

HTTP request là gói tin chứa các thông điệp được Client gửi đi để bắt đầu một hành động trên máy chủ. Start line của gói tin request chứa ba yếu tố:

Một HTTP method (như GET, PUT hoặc POST) mô tả hành động sẽ được thực hiện. Ví dụ: GET cho biết tài nguyên cần được tìm nạp. POST có nghĩa là dữ liệu được đẩy đến máy chủ (tạo hoặc sửa đổi tài nguyên hoặc tạo tài liệu tạm thời để gửi lại).

Request target, thường là URL hoặc đường dẫn tuyệt đối của giao thức, port và domain, thường được đặc trưng bởi hoàn cảnh yêu cầu. Định dạng của request target này khác nhau giữa các phương thức HTTP khác nhau. Nó có thể là:

Một đường dẫn tuyệt đối, theo sau cùng là một dấu '?' và một chuỗi truy vấn. Đây là dạng phổ biến nhất, được gọi là dạng gốc và được sử dụng với các phương thức GET, POST, HEAD và OPTIONS.

- POST / HTTP/1.1
- GET /background.png HTTP/1.0
- HEAD /test.html?query=alibaba HTTP/1.1

- OPTIONS /anypage.html HTTP/1.0

URL hoàn chỉnh (dạng tuyệt đối) chủ yếu được sử dụng với GET khi kết nối với proxy.

GET https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages HTTP/1.1

Thành phần quyền hạn của một URL, bao gồm tên miền và tùy chọn cổng (có tiền tố là một dấu ':'), được gọi là biểu mẫu quyền hạn . Nó chỉ được sử dụng với CONNECT khi thiết lập đường hầm.

HTTP. CONNECT developer.mozilla.org:80 HTTP/1.1

Dạng dấu hoa thị , một dấu hoa thị đơn giản ('*') được sử dụng cùng với OPTIONS, đại diện toàn bộ máy chủ. OPTIONS * HTTP/1.1

Phiên bản HTTP xác định cấu trúc của thông báo còn lại, hoạt động như là một chỉ báo về phiên bản dự kiến sẽ sử dụng cho phản hồi.

1.4.1.2 Header

HTTP header của một gói tin request tuân theo cấu trúc cơ bản giống như HTTP header: một chuỗi không phân biệt chữ hoa chữ thường, theo sau là dấu hai chấm (':') và một giá trị có cấu trúc phụ thuộc vào header. Toàn bộ header chỉ gồm một dòng duy nhất.

```
POST / HTTP/1.1
```

Host: localhost:8080 User-Agent: Mozilla /5.0 (Macintosh;…)… Firefox/51.0 Accept: text/html,application/xhtml+xml,…,*/*;q=0.8 Accept-Language: en-US, en;q=0.5 Accept-Encoding: gzip, deflate	Request headers
Connection: keep-alive Upgrade-Insecure-Request: 1	General headers
Content-Type: multipart/form-data; boundary= -12656974 Content-Length: 345	Representation headers

Hình 1. 5: Cấu trúc gói tin HTTP Request

Các phương thức HTTP:

- **Phương thức GET:** Get là phương thức được Client gửi dữ liệu lên Server thông qua đường dẫn URL nằm trên thanh địa chỉ của Browser. Server sẽ nhận đường dẫn đó và phân tích trả về kết quả cho bạn. Hơn nữa, nó là một phương thức được sử dụng phổ biến mà không cần có request body.
- **Phương thức POST:** Post là phương thức gửi dữ liệu đến server giúp bạn có thể thêm mới dữ liệu hoặc cập nhật dữ liệu đã có vào database.
- **Phương thức PUT:** dùng để cập nhật dữ liệu đã có sẵn trong database.
- **Phương thức DELETE:** Delete sẽ xoá các dữ liệu của server về tài nguyên thông qua URL

- **Phương thức HEAD:** HEAD giống với GET nhưng phản hồi sẽ không nhận được response body.

1.4.1.3 Body

Phần cuối cùng của gói tin request là phần body. Không phải tất cả request đều chứa các yêu cầu tìm nạp tài nguyên, như GET, HEAD, DELETE hoặc OPTIONS, thường không cần một tài nguyên. Một số yêu cầu gửi dữ liệu đến máy chủ để cập nhật dữ liệu: thường xảy ra với POST các yêu cầu (chứa dữ liệu biểu mẫu HTML).

Phần body có thể được chia thành hai loại:

- Đơn tài nguyên, bao gồm một tệp duy nhất, được xác định bởi hai header: Content-Type và Content-Length.
- Đa tài nguyên, gồm nhiều phần body, mỗi phần chứa một bit thông tin khác nhau. Điều này thường được liên kết với HTML Form.

1.4.2 Cấu trúc gói tin HTTP Response

1.4.2.1 Status line

Start line của HTTP response, được gọi là status line, chứa thông tin sau:

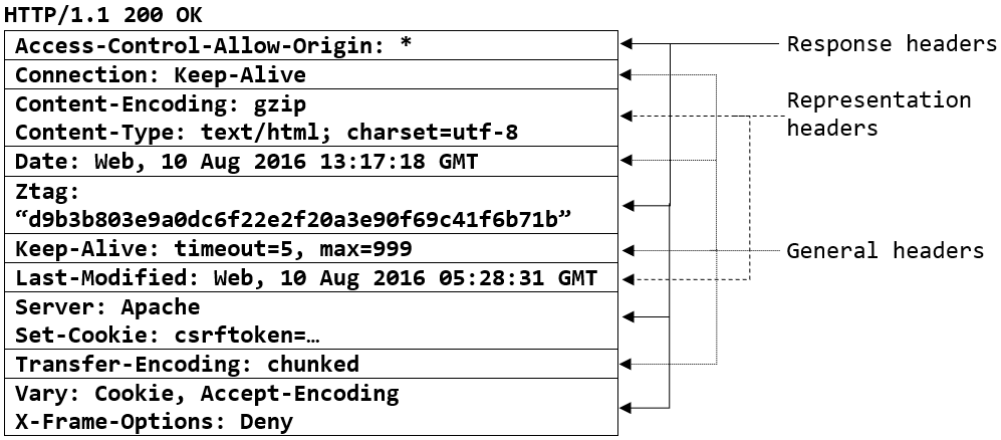
- Phiên bản giao thức, thường là HTTP/1.1.
- Status code: cho biết yêu cầu thành công hay thất bại. Mã trạng thái phổ biến là 200, 404, hoặc 302.
- Status text: Một mô tả ngắn gọn bằng văn bản về status code để giúp con người hiểu được.

Ý nghĩa của các Status code trong HTTP:

- **1xx (100 – 199): Information responses / Phản hồi thông tin** – Yêu cầu đã được chấp nhận và quá trình xử lý yêu cầu của bạn đang được tiếp tục.
- **2xx (200 – 299): Successful responses / Phản hồi thành công** – Yêu cầu của bạn đã được máy chủ tiếp nhận, hiểu và xử lý thành công.
- **3xx (300 – 399): Redirects / Điều hướng** – Phía client cần thực hiện hành động bổ sung để hoàn tất yêu cầu.
- **4xx (400 – 499): Client errors / Lỗi phía client** – Yêu cầu không thể hoàn tất hoặc yêu cầu chứa cú pháp không chính xác. 4xx sẽ hiện ra khi có lỗi từ phía client do không đưa ra yêu cầu hợp lệ.
- **5xx (500 – 599): Server errors / Lỗi phía máy chủ** – Máy chủ không thể hoàn thành yêu cầu được cho là hợp lệ. Khi 5xx xảy ra, bạn chỉ có thể đợi để bên hệ thống máy chủ xử lý.

1.4.2.2 Header

Phần header của gói tin response tuân theo cấu trúc giống như HTTP header: chuỗi không phân biệt chữ hoa chữ thường, theo sau là dấu hai chấm (':') và giá trị có cấu trúc phụ thuộc vào loại tiêu đề. Toàn bộ header chỉ gồm một dòng duy nhất.



Hình 1. 6: Cấu trúc gói tin HTTP Response

1.4.2.3 Body

Phần cuối cùng của gói tin response là phần body, có thể được chia thành ba loại:

- Đơn tài nguyên gồm một tệp có độ dài đã biết, được xác định bởi hai header: Content-Type và Content-Length.
- Đơn tài nguyên gồm một tệp duy nhất có độ dài không xác định, được mã hóa theo các đoạn có Transfer-Encoding đặt thành chunked.
- Đa tài nguyên , gồm nhiều phần trong body, mỗi phần chứa một phần thông tin khác nhau. Tuy nhiên loại này khá hiếm.

1.5 Phiên bản HTTP/2

1.5.1 Giới thiệu tổng quát về HTTP/2

HTTP/2 là tên gọi chính thức cho phiên bản tiếp theo của HTTP dựa trên công nghệ lõi từ SPDY (speedy) được phát triển bởi Google.

Tính ưu việt từ SPDY đã thu hút rất nhiều sự quan tâm từ các nhà phát triển giao thức và trình duyệt web hàng đầu. Vào thời điểm này, nhóm phát triển SPDY cho biết họ đang chuẩn hoá giao thức. Các bên đã đi đến thống nhất là phát triển hẳn lên HTTP/2 dựa trên SPDY.

Một số vấn đề từ HTTP/1 mà SPDY ra đời để giải quyết:

- Single request per connection. HTTP chỉ có thể tải về từng resource với từng request, mỗi cái là một connection tương ứng. Việc này gây làm tăng

delay (thời gian chờ) cũng như không tận dụng được băng thông. SPDY cho phép một connection có thể xử lý được đồng thời nhiều request.

- Client-initiated requests. HTTP hoạt động theo kiểu "hỏi-đáp", client request dữ liệu nào thì được server trả về dữ liệu nấy. Với SPDY, server có thể chủ động "push" dữ liệu về client mà không cần client gởi request đến.
- Redundant headers. HTTP headers là những metadata mô tả cách thức vận hành của một request HTTP. Trong nhiều trường hợp, rất nhiều request với những header lập đi lập lại giống nhau. SPDY có thể loại bỏ những header không cần thiết này để giải lượng băng thông cần thiết.
- Uncompressed data. Compression (nén) sẽ giúp dữ liệu có dung lượng nhỏ hơn khi trao đổi thông tin, với HTTP thì nó là optional (không bắt buộc). SPDY bắt buộc sử dụng nén dữ liệu cho mọi request.

1.5.2 HTTP Frames

HTTP/2 sử dụng frames để gửi dữ liệu giữa client và server. Mỗi loại frame có cấu trúc và mục đích riêng biệt. Dưới đây là một cấu trúc tổng quan của một HTTP/2 frame:

- Length (24 bits): Xác định độ dài của toàn bộ frame, không bao gồm phần đầu (length field) của frame. Độ dài có giới hạn tối đa là 16,777,215 bytes.
- Type (8 bits): Xác định loại của frame, chẳng hạn như DATA, HEADERS, SETTINGS, PUSH_PROMISE, và nhiều loại frame khác.
- Flags (8 bits): Cờ (flags) cung cấp thông tin về frame và có thể điều chỉnh cách server và client xử lý nó. Các cờ có thể thay đổi theo từng loại frame.
- R (1 bit): Bit đầu tiên trong Stream Identifier, được gọi là R-bit, xác định xem frame có ưu tiên (R=1) hay không (R=0).
- Stream Identifier (31 bits): Định danh luồng (stream) mà frame này liên quan. Stream Identifier xác định luồng cụ thể mà frame được gửi và nhận.
- Frame Payload: Nội dung thực tế của frame, bao gồm dữ liệu liên quan đến loại frame cụ thể. Frame có thể có payload (dữ liệu) hoặc không tùy theo loại frame.

Các loại frame khác nhau sẽ có cấu trúc payload khác nhau. Ví dụ, frame DATA sẽ chứa dữ liệu tải về hoặc tải lên, trong khi frame HEADERS sẽ chứa thông tin tiêu đề của yêu cầu hoặc phản hồi.

HTTP/2 sử dụng các loại frame này để quản lý và điều khiển giao tiếp giữa client và server, cải thiện hiệu suất và tốc độ truyền dữ liệu trên mạng.

1.5.3 Streams

HTTP/2 sử dụng các "streams" để quản lý và đồng bộ hóa dữ liệu giữa máy khách và máy chủ. Một stream là một luồng dữ liệu độc lập trên kết nối HTTP/2. Dưới đây là một số điểm quan trọng về streams trong HTTP/2:

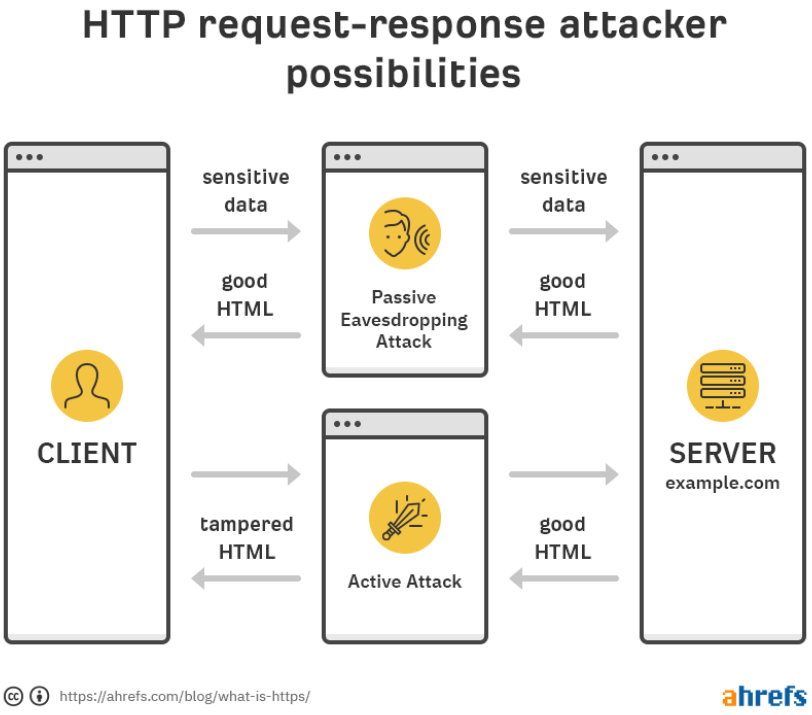
- Stream Identifier: Mỗi stream được xác định bằng một Stream Identifier, là một số nguyên 31 bit không âm. Bit đầu tiên (R-bit) trong Stream Identifier xác định xem stream có ưu tiên (R=1) hay không (R=0).
- Multiplexing: HTTP/2 hỗ trợ multiplexing, cho phép nhiều streams được gửi qua cùng một kết nối. Điều này giúp tối ưu hóa sử dụng kết nối và cải thiện hiệu suất, vì các streams có thể hoạt động độc lập và không phải chờ đợi lẫn nhau.
- Prioritization: HTTP/2 cho phép ưu tiên hóa các streams, giúp xác định xem stream nào quan trọng hơn và cần được xử lý trước. Prioritization giúp tối ưu hóa việc tải tài liệu và đảm bảo rằng các tài liệu quan trọng được tải trước.
- Dependent Streams: HTTP/2 cho phép xác định sự phụ thuộc giữa các streams, cho phép tải các tài liệu phụ thuộc trước khi tải tài liệu chính. Điều này giúp tối ưu hóa thời gian tải trang web.
- Flow Control: Mỗi stream trong HTTP/2 có cửa sổ trượt (flow control window) để kiểm soát lưu lượng dữ liệu. Flow control giúp ngăn chặn quá tải máy khách hoặc máy chủ bằng cách quản lý kích thước của dữ liệu được gửi.
- Server Push: Server push là một tính năng trong HTTP/2 cho phép máy chủ trình bày trước các tài liệu cho máy khách mà máy khách chưa yêu cầu. Điều này giúp tối ưu hóa tải trang web bằng cách loại bỏ cần phải làm lại nhiều yêu cầu từ máy khách.

Streams trong HTTP/2 là một phần quan trọng trong việc cải thiện hiệu suất và tốc độ tải trang web trên mạng. Chúng giúp tối ưu hóa sử dụng kết nối, đảm bảo ưu tiên cho các tài liệu quan trọng, và cung cấp cách hiệu quả để quản lý và đồng bộ hóa dữ liệu giữa máy khách và máy chủ.

CHƯƠNG II: TỔNG QUAN VỀ GIAO THỨC HTTPS

2.1 Sự cần thiết của HTTPS

Nguy cơ bị tấn công mạng có thể xảy ra bất cứ đâu với những thiết bị mạng thiếu an toàn, nhất là việc sử dụng wifi công cộng. Hiện nay, nhiều người đã nhận thức được rủi ro này nên đã sử dụng VPN. Tuy nhiên, việc sử dụng dịch vụ website an toàn lại hoàn toàn thuộc về trách nhiệm của người quản trị Web.



Hình 2. 1: Các khả năng tấn công HTTP Response

Hình ảnh trên mô phỏng những gì xảy ra khi client tương tác với server qua giao thức HTTP thông thường. Kẻ tấn công có thể xem những dữ liệu nhạy cảm như thông tin đăng nhập, thông tin về những thanh toán, hoặc có thể tiêm mã độc vào những requested. Vì vậy snên HTTPS (Hypertext Transfer Protocol Security) đã ra đời để có thể ngăn chặn những hành vi tấn công trên.

2.2 Giới thiệu về HTTPS

Giao thức HTTPS là gì? Thực chất giao thức HTTPS là phiên bản mã HTTP tích hợp thêm chứng chỉ bảo mật SSL, TLS nhằm mã hóa các thông điệp để tăng tính bảo mật. Từ đó kẻ tấn công chỉ có thể tiếp cận được những chuỗi kí tự mã hóa thay vì những thông tin nhạy cảm (thẻ ngân hàng, tài khoản mật khẩu, tin nhắn riêng tư,…).

HTTPS hoạt động cơ bản dựa trên hệ thống PKI (Public Key Infrastructure – khóa công khai) không đối xứng. Hệ thống này sử dụng một cặp khóa để mã hóa thông tin liên lạc. Bất cứ thông tin nào được mã hóa bằng khóa công khai chỉ có thể được giải mã bằng khóa riêng và ngược lại.

2.2.1 SSL/TLS (Secure Socket Layer/Transport Layer Security)

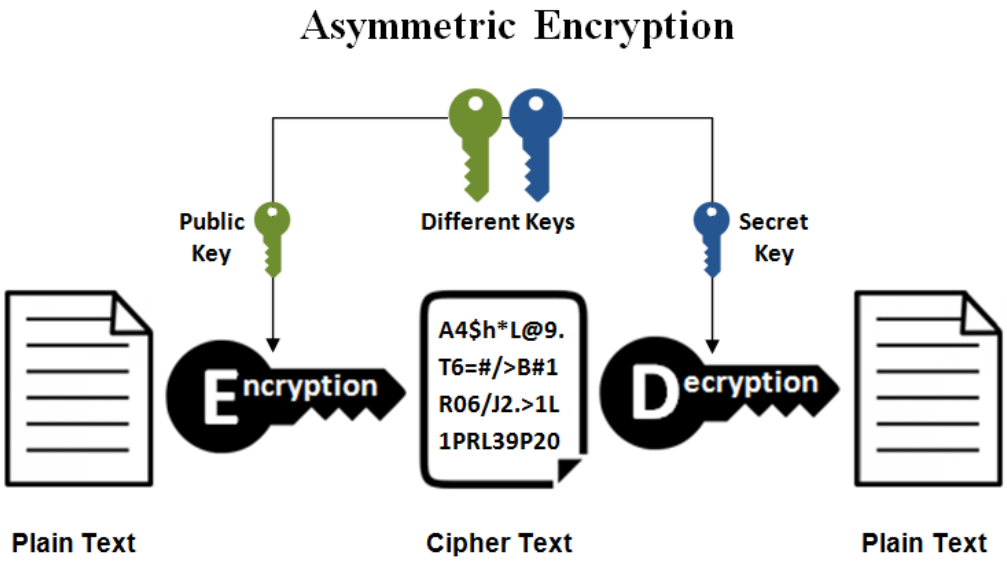
2.2.1.1 Cách SSL/TLS hoạt động

Về cơ bản SSL/TLS hoạt động theo 2 loại chính là Asymmetric Cryptography (Mã hóa bất đối xứng) và Symmetric Cryptography (Mã hóa đối xứng).

- **Mã hóa bất đối xứng (hay còn gọi là mã khóa công khai)**

Mã hóa bất đối xứng sử dụng một hàm toán học để sinh cặp khóa mã hóa và giải mã dữ liệu. Trong cặp khóa này, một khóa được chia sẻ với bất cứ ai muốn tham gia giao tiếp. Đây gọi là khóa công khai. Một chiếc khóa còn lại được giữ để giải mã thông điệp. Gọi là khóa bí mật.

Trong mã hóa bất đối xứng, người gửi mã hóa thông điệp bằng khóa công khai (của người nhận) rồi gửi đi. Người nhận giải mã tin này bằng khóa bí mật còn lại trong cặp khóa.

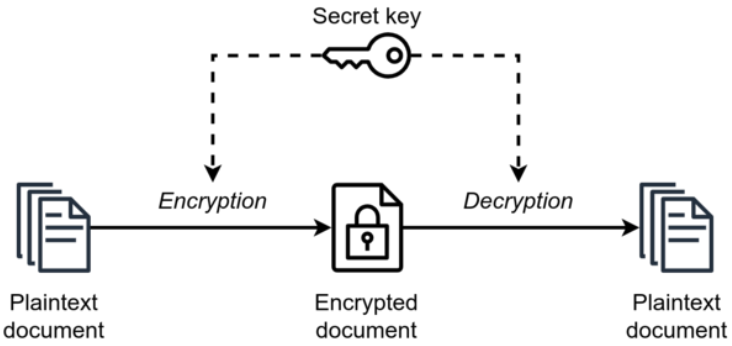


Hình 2. 2: Asymmetric Encryption

Giao tiếp qua SSL luôn bắt đầu bằng SSL Handshake, nó cho phép browser xác minh máy chủ Web, lấy khóa công khai và thiết lập kết nối an toàn trước khi bắt đầu truyền dữ liệu thực tế.

- **Mã hóa đối xứng (Symmetric Cryptography)**

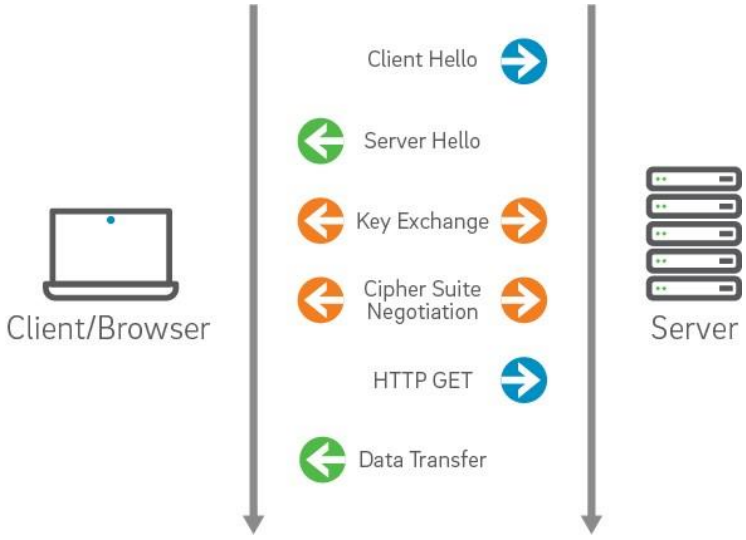
Trong mã đối xứng, chỉ có duy nhất một khóa làm công việc mã hóa và giải mã dữ liệu. Chỉ người gửi và người nhận có khóa này.



Hình 2. 3: Symmetric Encryption

SSL/TLS sử dụng mã đối xứng để tạo phiên sau khi quá trình SSL/TLS Handshake thành công. Thuật toán thường được dùng trong mã đối xứng là AES-128, AES-192 và AES-256.

2.2.1.2 Các bước truyền dữ liệu qua SSL/TLS



Hình 2. 4: Mô hình giao thức HTTPS

Bước 1: Đầu tiên, client gửi một message Client Hello tới server. Message này bao gồm version của SSL trên client, cipher setting (các thiết lập về mật mã), session data và các thông tin cần thiết khác mà server cần để client có thể giao tiếp thông qua giao thức SSL.

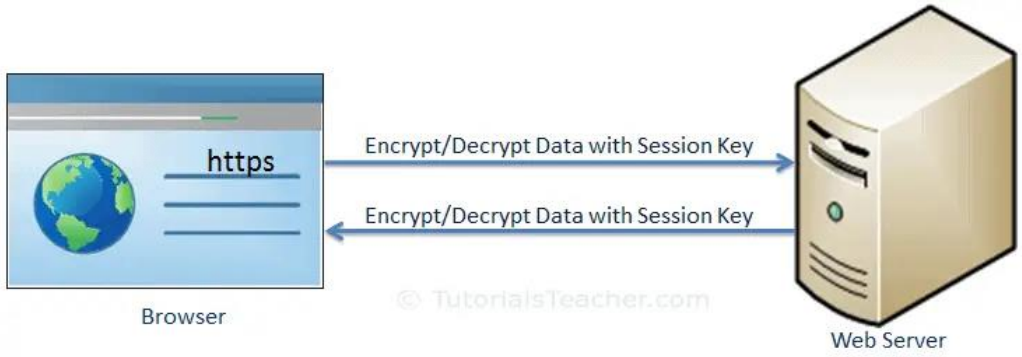
Bước 2: Sau khi nhận được gói tin từ client, server phản hồi với một message Server Hello. Message này cũng bao gồm version của SSL trên server, thiết lập mật mã, session data, public key và các thông tin cần thiết khác mà client cần để giao tiếp thông qua SSL.

Bước 3: Khi đã thống nhất được loại mã hóa và các thông tin cần thiết, client tiến hành xác nhận SSL certificate (public key và các thông tin khác mà server vừa gửi) với Certificate Authority - CA (là các đơn vị thứ 3 cung cấp dịch vụ chứng thực số, ví dụ: GeoTrust, Digicert...) xem SSL certificate đó có là "đồ thật" - không bị giả mạo hay không. Nếu xác nhận thất bại thì client sẽ từ chối giao tiếp, quá trình bắt tay SSL sẽ bị dừng lại. Nếu xác nhận thành công thì tiếp tục.

Bước 4: Sau khi xong, client sinh một session key, mã hóa nó với public key và gửi nó đến server.

Bước 5: Server dùng private key để giải mã session key sau đó gửi kết quả thành công về cho client. Kết này cũng sẽ được mã hóa đối xứng với session key mà vừa được giải mã.

Kết quả cuối cùng của quá trình "bắt tay" SSL, đó là cả client và server đều có trong tay session key, khóa này về sau sẽ được dùng để mã hóa và giải mã các data thực tế cần truyền tãi giữa client và server. Từ lúc này trở đi cặp khóa public key và private key sẽ không được sử dụng nữa.



Hình 2. 5: Quá trình SSL Handshake

2.2.2 Chứng chỉ SSL

Chứng chỉ SSL là tập tin được cấp bởi cơ quan có thẩm quyền. Như đã biết, SSL sử dụng mã hóa bất đối xứng để mã hóa liên kết giữa hai hệ thống sử dụng một cặp khóa (khóa công khai và khóa bí mật). Chứng chỉ SSL chứa khóa công khai của chủ sở hữu và các thành phần khác. Web server gửi khóa công khai cho browser xác minh và xác thực xem trang web này có chứng chỉ SSL hay không.

Các bước để có được chứng chỉ SSL:

Bước 1: Tạo yêu cầu chứng chỉ SSL (CSR - Certificate Signing Request): Trước khi bạn có thể có một chứng chỉ SSL cho máy chủ của mình, bạn cần tạo một CSR. CSR chứa thông tin về bạn hoặc tổ chức của bạn và khóa công cộng mà bạn sẽ sử dụng cho chứng chỉ SSL.

Bước 2: Gửi CSR đến CA (Certificate Authority): Sau khi bạn đã tạo CSR, bạn cần gửi nó đến một CA tin cậy. CA là một tổ chức hoặc dịch vụ chuyên cung cấp chứng chỉ SSL. CA sẽ kiểm tra thông tin trong CSR và sau đó tạo một chứng chỉ SSL với thông tin này.

Bước 3: Tạo chữ ký số cho chứng chỉ SSL: CA sẽ sử dụng khóa riêng (private key) của họ để ký chứng chỉ SSL của bạn. Điều này tạo ra chữ ký số, là một phần của chứng chỉ SSL, để xác minh tính xác thực của chứng chỉ.

Bước 4: Phân phối chứng chỉ SSL và chữ ký số: Sau khi chứng chỉ SSL đã được ký kết, CA sẽ gửi lại chứng chỉ này cho bạn. Bạn sau đó cần cài đặt chứng chỉ SSL và khóa riêng (private key) của mình trên máy chủ web của bạn.

Chữ ký số: có chức năng tương tự như chữ ký tay, nó là cách xác minh online của người gửi. Chữ ký số có 3 ứng dụng trong trang web:

- Xác thực (Authentication): Chữ ký số cho người nhận lý do để tin vào dữ liệu được tạo và chuyển đi từ người gửi. Đối với trang web, chữ ký số giúp Client xác thực Web server không bị giả mạo.
- Tính chống chối bỏ (Non-repudiation): Tính năng chống chối bỏ giúp bảo vệ trang web khỏi các tình huống người dùng hoặc máy chủ phủ nhận việc thực hiện giao dịch hoặc trao đổi thông tin. Khi có một chữ ký số, không thể đơn giản từ chối hoặc phủ nhận việc thực hiện hành động nào đó trên trang web.
- Tính toàn vẹn (Integrity): Chữ ký số đảm bảo rằng dữ liệu không bị thay đổi trên đường truyền.

2.3 Quy trình xác thực và mã hóa trong HTTPS

HTTPS (Hypertext Transfer Protocol Secure) là một phiên bản bảo mật của giao thức HTTP được sử dụng để truyền tải dữ liệu trên Internet. Quá trình xác thực và mã hóa trong HTTPS giúp đảm bảo tính bảo mật và an toàn của dữ liệu truyền qua mạng.

Quy trình tổng quan về cách HTTPS thực hiện xác thực và mã hóa:

Yêu cầu HTTPS: Quá trình bắt đầu khi người dùng nhập một URL bắt đầu bằng "https://" vào trình duyệt hoặc khi một ứng dụng hoặc máy chủ yêu cầu kết nối bảo mật.

Giao tiếp với máy chủ: Trình duyệt hoặc ứng dụng sẽ tạo một kết nối đến máy chủ web bằng cách sử dụng giao thức SSL/TLS (Secure Sockets Layer/Transport Layer Security).

Sử dụng giao thức SSL/TLS: SSL/TLS sẽ thực hiện quá trình xác thực và mã hóa dữ liệu theo cách sau:

- **Xác thực máy chủ (Server Authentication):** Máy chủ sẽ gửi một chứng chỉ SSL/TLS kèm theo dữ liệu công khai (public key) của nó cho browser hoặc app. Browser sẽ kiểm tra chứng chỉ này để xác minh tính hợp pháp của máy chủ. Chứng chỉ này thường được cấp bởi một cơ quan chứng thực (Certificate Authority - CA) đáng tin cậy.
- **Xác thực ngược (Reverse Authentication):** Trình duyệt hoặc ứng dụng có thể yêu cầu máy chủ xác minh tính hợp pháp của nó bằng cách gửi một chứng chỉ client (chứng chỉ SSL/TLS) cùng với dữ liệu công khai của nó. Tùy thuộc vào cài đặt cụ thể, máy chủ có thể yêu cầu hoặc bỏ qua bước này.
- **Khởi tạo phiên mã hóa (Encryption Session Establishment):** Sau khi xác thực thành công, máy chủ và trình duyệt hoặc ứng dụng sẽ thỏa thuận trên một phiên mã hóa đối tượng (session key). Phiên mã hóa này sẽ được sử dụng để mã hóa và giải mã dữ liệu truyền qua mạng.

Truyền tải dữ liệu mã hóa: Dữ liệu giữa máy chủ và trình duyệt hoặc ứng dụng sẽ được mã hóa bằng cách sử dụng phiên mã hóa đã thỏa thuận. Điều này đảm bảo rằng bất kỳ thông tin nào cũng không thể bị đọc hoặc hiểu được nếu nó bị chặn bởi các kẻ tấn công.

Kết thúc phiên kết nối: Sau khi phiên kết nối hoàn tất hoặc sau một khoảng thời gian cố định, phiên kết nối sẽ được đóng, và trình duyệt hoặc ứng dụng sẽ chấp nhận một phiên kết nối mới nếu cần.

Quá trình này đảm bảo rằng dữ liệu truyền qua mạng được bảo vệ khỏi các nguy cơ như đánh cắp thông tin, đánh cắp danh tính và sửa đổi dữ liệu.

2.4 Cách chuyển đổi từ HTTP sang HTTPS

Cài đặt chứng chỉ trên máy chủ web: Sau khi có chứng chỉ SSL/TLS, bạn cần cài đặt nó trên máy chủ web của bạn. Quy trình này sẽ khác nhau tùy thuộc vào loại máy chủ bạn đang sử dụng (ví dụ: Apache, Nginx, IIS). Hầu hết các tổ chức cấp chứng chỉ sẽ cung cấp hướng dẫn cài đặt cho máy chủ cụ thể của bạn.

Cấu hình máy chủ web: Sau khi cài đặt chứng chỉ, bạn cần cấu hình máy chủ web để sử dụng HTTPS thay vì HTTP. Điều này bao gồm việc sửa đổi tệp cấu hình của máy chủ để ánh xạ các yêu cầu HTTPS đến thư mục và tệp tương ứng. Bạn cũng cần xác định cổng (thông thường là cổng 443) để máy chủ lắng nghe các yêu cầu HTTPS.

Sửa các liên kết và tài liệu tĩnh: Nếu bạn sử dụng tài liệu tĩnh hoặc có các liên kết cứng mã hóa HTTP, bạn cần sửa đổi chúng để sử dụng HTTPS thay vì HTTP. Điều này đảm bảo rằng các trang web của bạn sẽ hiển thị chính xác và an toàn.

Kiểm tra và cập nhật ứng dụng web: Nếu bạn có một ứng dụng web, bạn cần kiểm tra xem ứng dụng có hỗ trợ HTTPS không. Nếu có, bạn cần cập nhật cài đặt của ứng dụng để sử dụng HTTPS. Điều này đảm bảo rằng các tính năng và chức năng của ứng dụng hoạt động đúng cách.

Kiểm tra và cấu hình lại máy chủ DNS: Cuối cùng, bạn cần kiểm tra và cấu hình lại các bản ghi DNS để ánh xạ tên miền của bạn (URL) đến địa chỉ IP của máy chủ web của bạn.

Khi bạn đã thực hiện các bước này, trang web của bạn sẽ sử dụng giao thức HTTPS thay vì HTTP và sẽ được mã hóa để đảm bảo tính bảo mật của người dùng. Điều này quan trọng đặc biệt khi bạn thu thập thông tin cá nhân hoặc tài liệu quan trọng trên trang web của mình.

2.5 Những ưu điểm của HTTPS đối với quyền bảo mật và riêng tư

Dữ liệu được mã hóa: HTTPS sử dụng mã hóa SSL/TLS để bảo vệ dự liệu truyền tải giữa máy tính người dùng và web server. Điều này ngăn chặn hacker tấn công từ việc đọc hiểu được dữ liệu gửi trên đường truyền. Tấn công man-in-the-middle sẽ không xảy ra, khi dùng HTTPS kẻ tấn công sẽ chỉ nhìn thấy những chuỗi kí tự ngẫu nhiên (dữ liệu được mã hóa) không có nghĩa.

Chứng thực máy chủ: HTTPS đảm bảo tính xác thực của máy chủ web. Khi truy cập một trang web, browser sẽ kiểm tra chứng chỉ SSL/TLS của máy chủ để xác minh tính xác thực của nó. Điều này giúp phòng tránh tấn công giả mạo, trong đó kẻ tấn công cố gắng làm giống trang web thật để lừa người dùng nhập thông tin vào.

Bảo vệ dữ liệu người dùng: HTTPS bảo vệ thông tin cá nhân của người dùng, chẳng hạn như tên người dùng, mật khẩu, thông tin tài khoản

ngân hàng và các dữ liệu nhạy cảm khác khỏi sự xâm nhập hoặc lấy cắp thông qua kết nối không an toàn.

2.6 Sự khác biệt giữa HTTP và HTTPS

	HTTP	HTTPS
Mã hóa dữ liệu	Dữ liệu truyền qua HTTP không được mã hóa, điều này có nghĩa thông tin gửi từ máy người dùng đến web server có thể bị đánh cắp hoặc theo dõi bởi kẻ tấn công.	Sử dụng mã hóa SSL/TLS để bảo vệ dữ liệu trong quá trình truyền tải. Điều này đảm bảo rằng thông tin không thể bị đọc được trên đường truyền.
Chứng thực máy chủ	Không cung cấp cơ chế kiểm tra tính xác thực của máy chủ web, do đó, trang web có thể dễ dàng bị tấn công giả mạo.	Sử dụng chứng chỉ SSL/TLS để xác thực máy chủ web. Trình duyệt kiểm tra chứng chỉ này để đảm bảo tính xác thực của trang web. Điều này giúp ngăn chặn tấn công giả mạo.
Tốc độ truy cập	Thường nhanh hơn so với HTTPS vì không có quá trình mã hóa dữ liệu.	Có thể chậm hơn một chút do quá trình mã hóa và giải mã, nhưng sự chậm trễ này đã giảm đi đáng kể trong các phiên bản gần đây của SSL/TLS.
Bảo vệ quyền riêng tư dữ liệu	Không cung cấp bảo vệ quyền riêng tư và dữ liệu của người dùng, dễ bị xâm nhập hoặc lấy cắp.	Cung cấp bảo vệ quyền riêng tư và dữ liệu, đặc biệt là khi người dùng gửi thông tin cá nhân như mật khẩu hoặc thông tin tài khoản ngân hàng.

CHƯƠNG 3: THỰC NGHIỆM PHÂN TÍCH GIAO THỨC

3.1 Phân tích giao thức HTTP

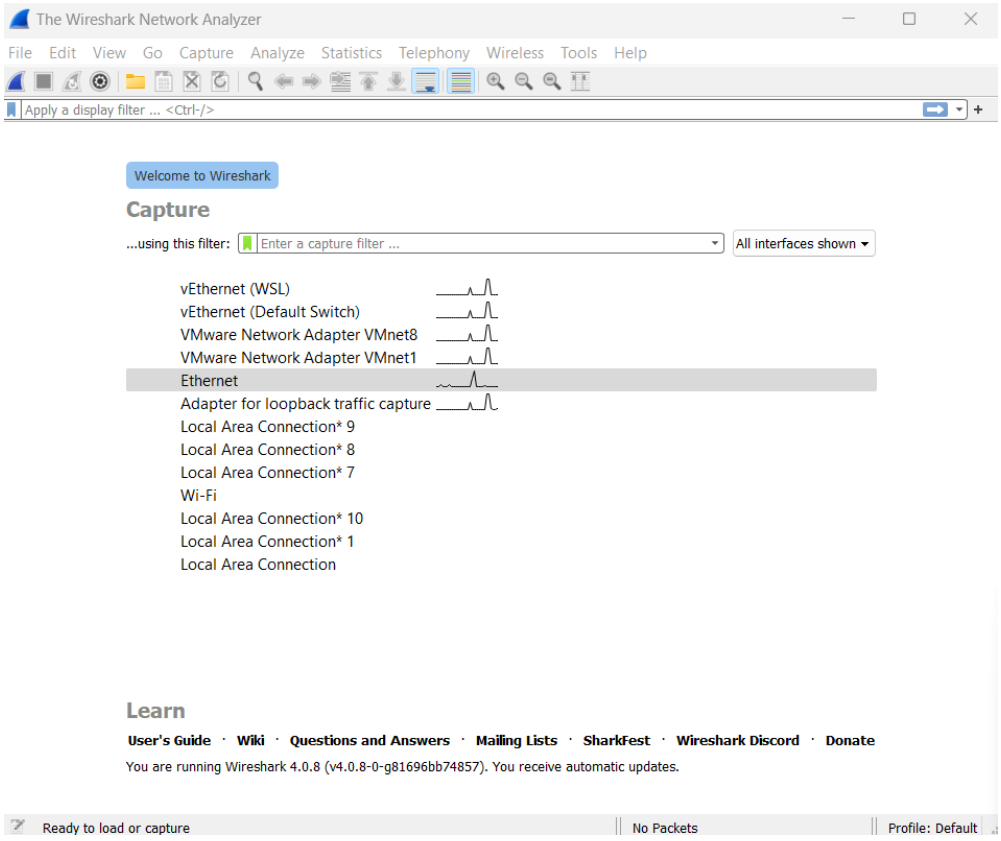
3.1.1 Mục tiêu

Việc phân tích gói tin HTTP để hiểu rõ cách hoạt động của giao thức này, cải thiện hiệu suất, đảm bảo bảo mật và quyền riêng tư, và tìm hiểu về cách ứng dụng của bạn tương tác với máy chủ và dữ liệu trên mạng.

3.1.2 Thực hiện

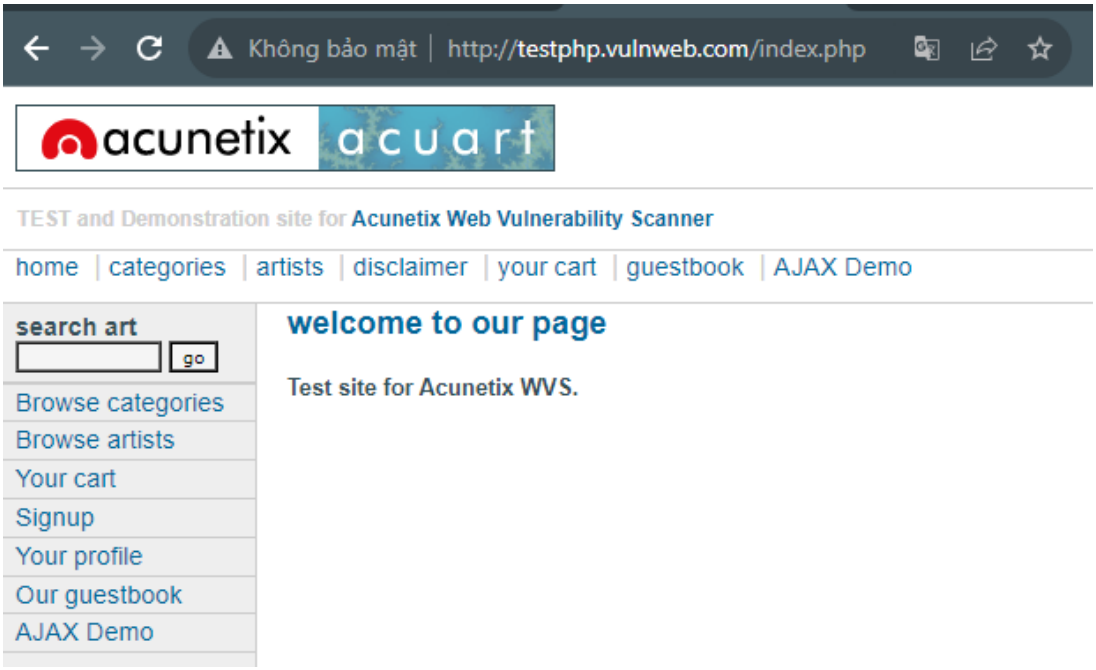
- **Bước 1: Chuẩn bị trước phân tích**

Sử dụng công cụ Wireshark, chọn phần Network interface thích hợp nhằm việc chọn đúng đường truyền mạng để có thể bắt được các gói tin mà client browser và web server trao đổi với nhau. Trong thực nghiệm này sẽ chọn network interface là ‘Ethernet’.

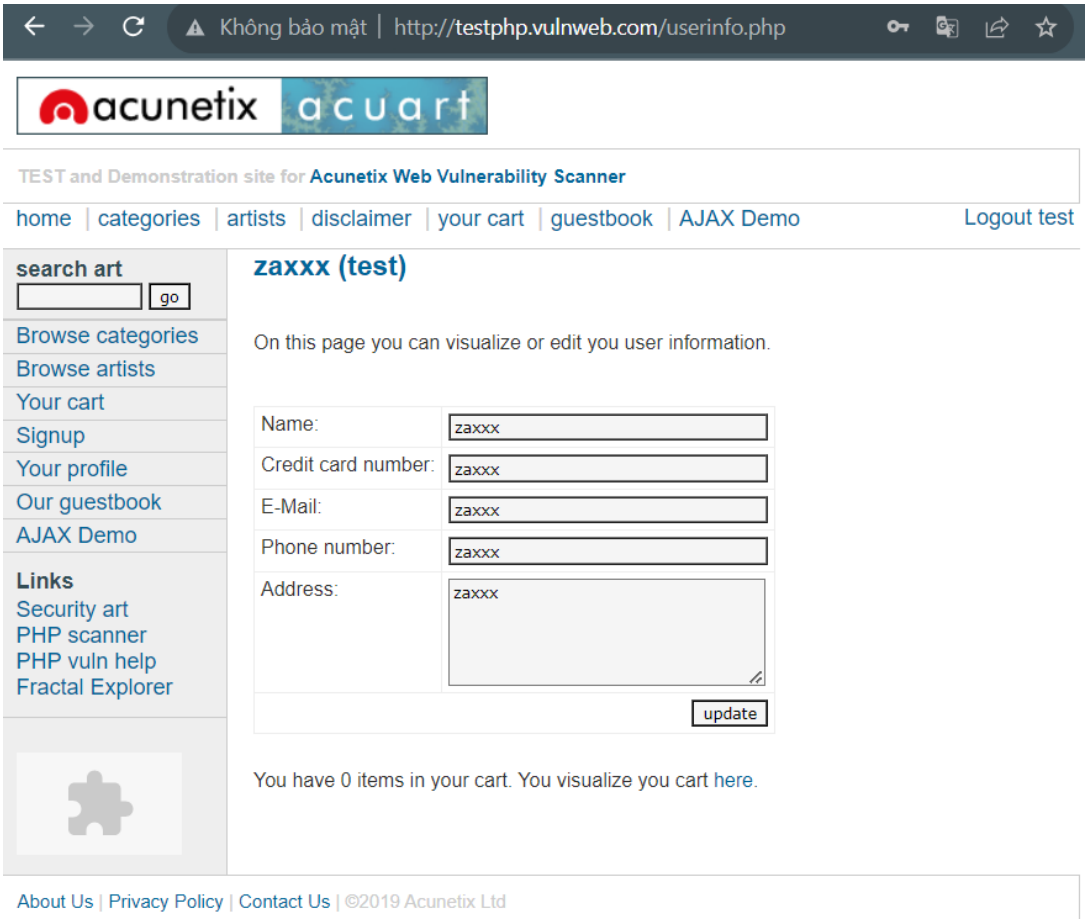


Hình 3. 1: Networks interface trong công cụ Wireshark

Sử dụng browser và truy cập vào một website sử dụng giao thức HTTP để có thể tiến hành quan sát các gói tin sử dụng giao thức HTTP. Trong thực nghiệm này sẽ sử dụng website có tên miền là ‘testphp.vulnweb.com’ sau đó nhấn vào Signup và nhập tên đăng nhập và mật khẩu là ‘test’ để tiến hành quan sát và phân tích các gói tin.

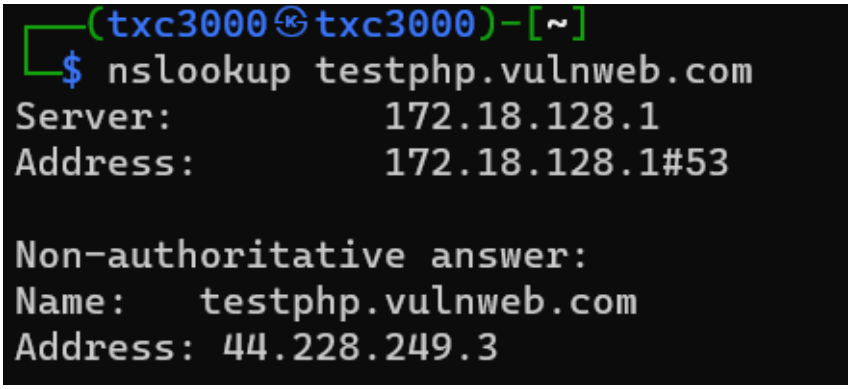


Hình 3. 2: Website testphp.vulnweb.com



Hình 3. 3: Website sau khi đăng nhập

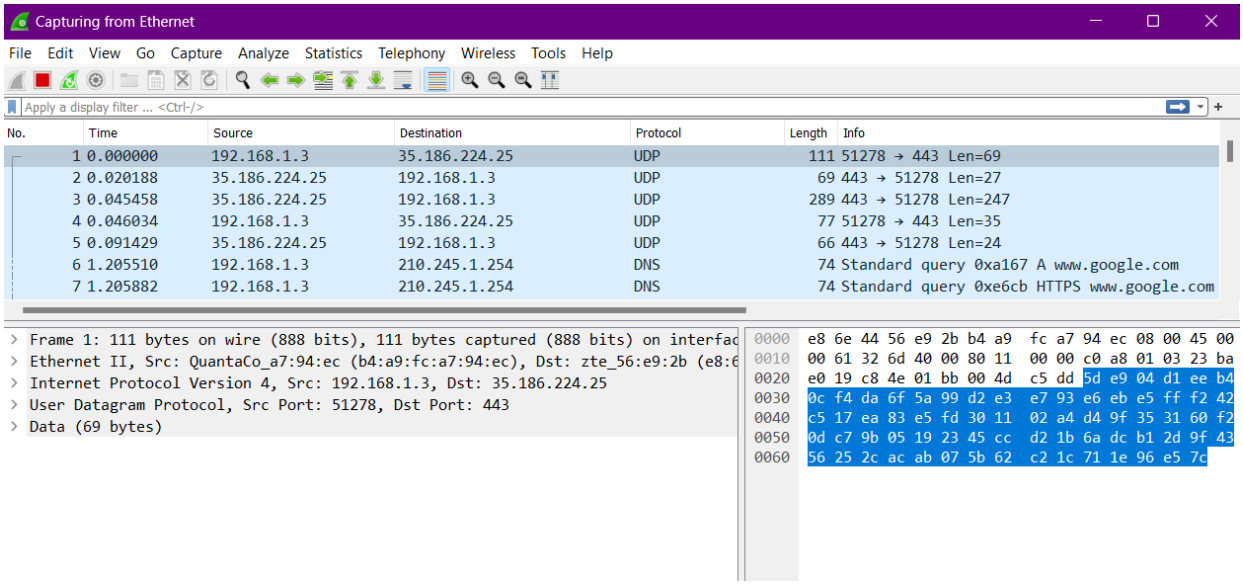
Ta có thể thấy website này sử dụng giao thức HTTP, tiến hành tìm địa chỉ IP của website bằng cách sử dụng lệnh ‘nslookup’ trên cmd.



Hình 3. 4: Địa chỉ IP của website testphp.vulnweb.com

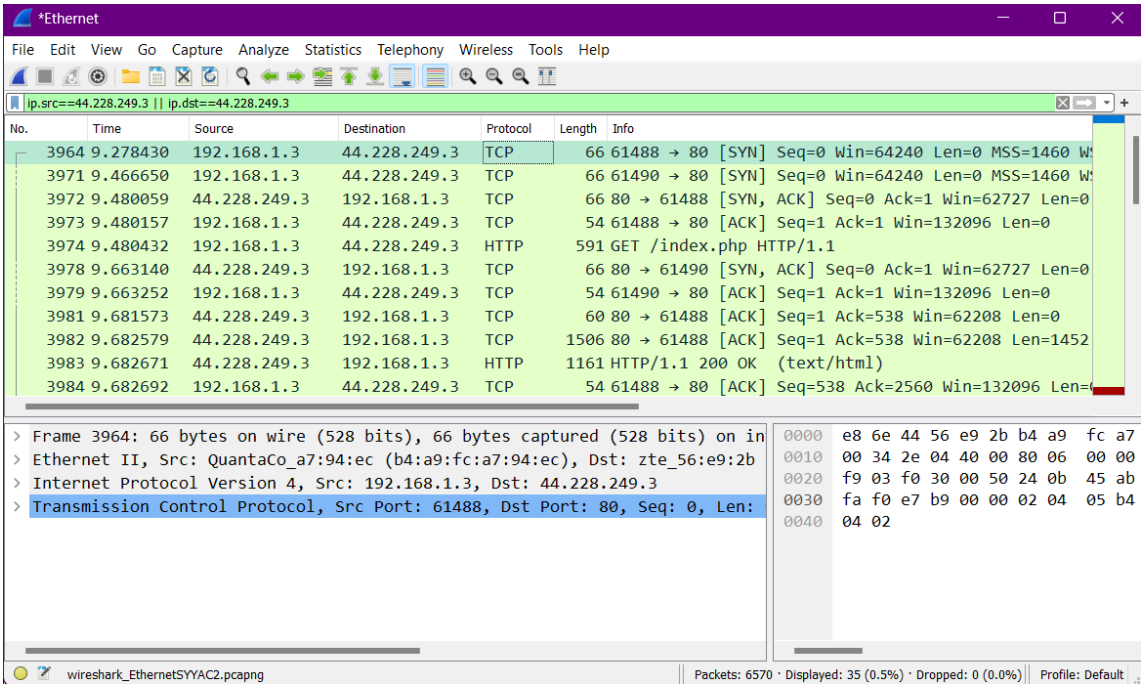
- **Bước 2: Phân tích gói tin**

Sau khi truy cập website trên browser, chuyển sang Wireshark, ta có thể thấy công cụ đã và đang bắt các gói tin giao tiếp giữa client browser và web server trên đường truyền là Ethernet.



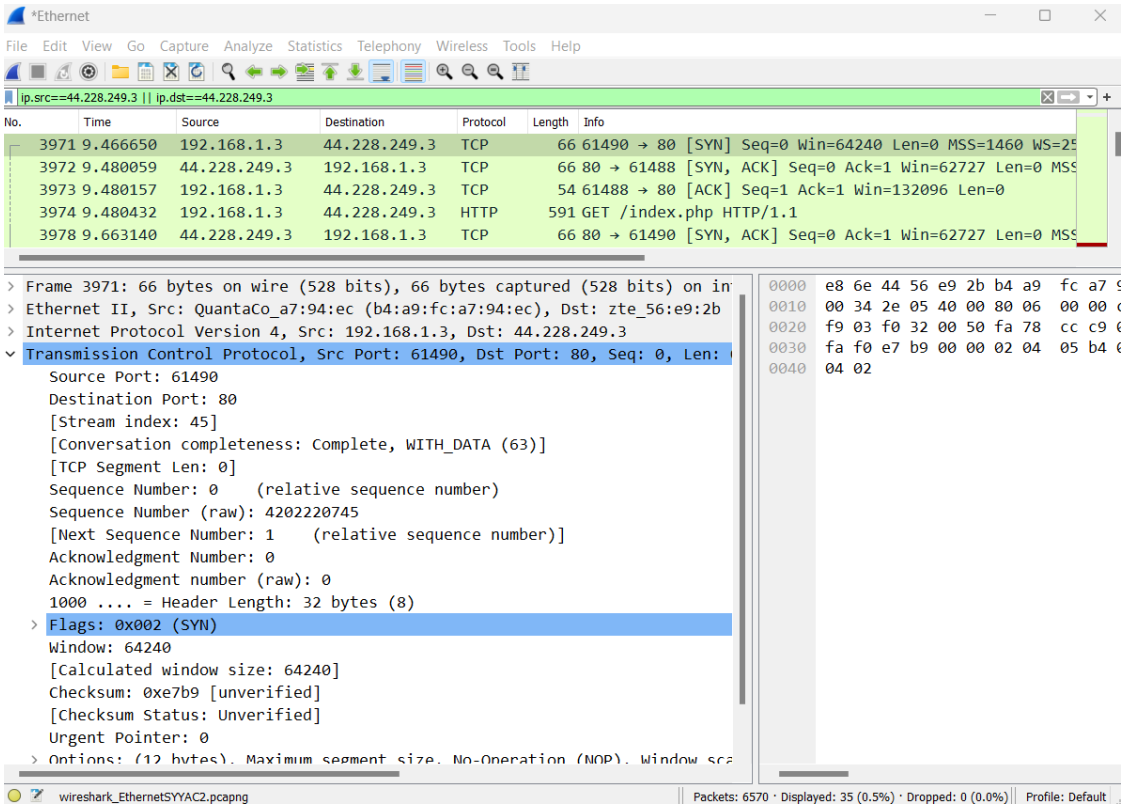
Hình 3. 5: Wireshark đã và đang bắt các gói tin

Tiến hành dừng chức năng bắt các gói tin của Wireshark và bắt đầu tiến hành lọc các gói tin có địa chỉ IP nguồn hoặc địa chỉ IP đích bằng địa chỉ IP của web server là 44.228.249.3 bằng cách nhập lệnh ‘ip.src==44.228.249.3 || ip.dst==44.228.249.3’ vào ô filter của công cụ rồi tìm kiếm.



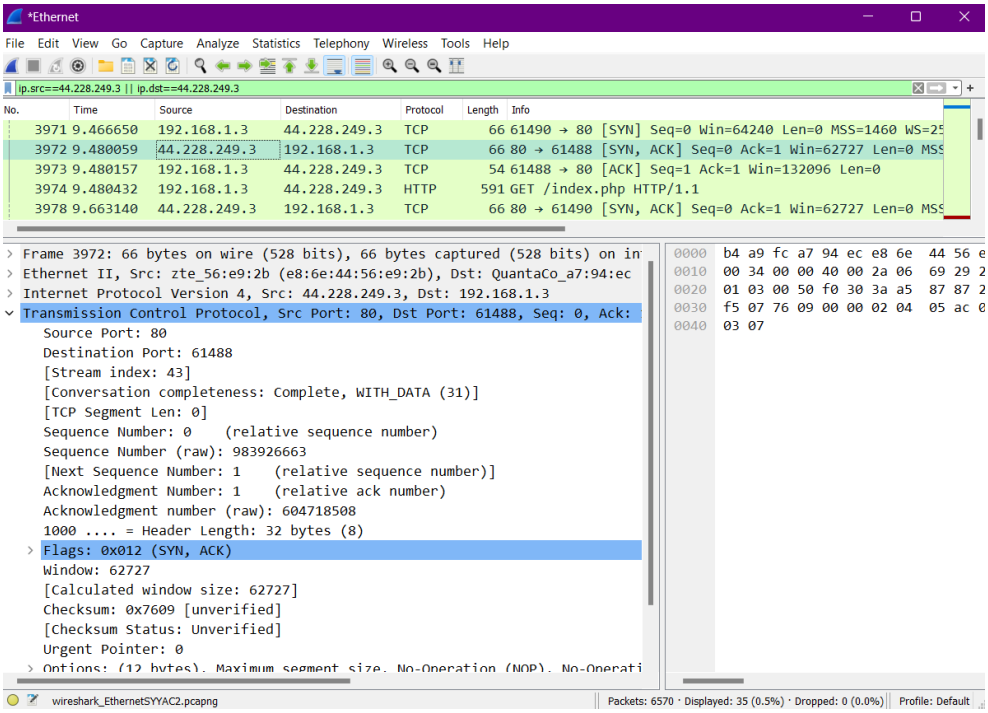
Hình 3. 6: Các gói tin giữa client browser và web server giao tiếp với nhau

Có thể thấy trước khi giao thức HTTP được sử dụng, client browser và web server đã thực hiện quá trình 'Bắt tay ba bước' để mở kết nối giữa 2 bên. Tiến hành phân tích từng gói tin trong quá trình bắt tay ba bước.



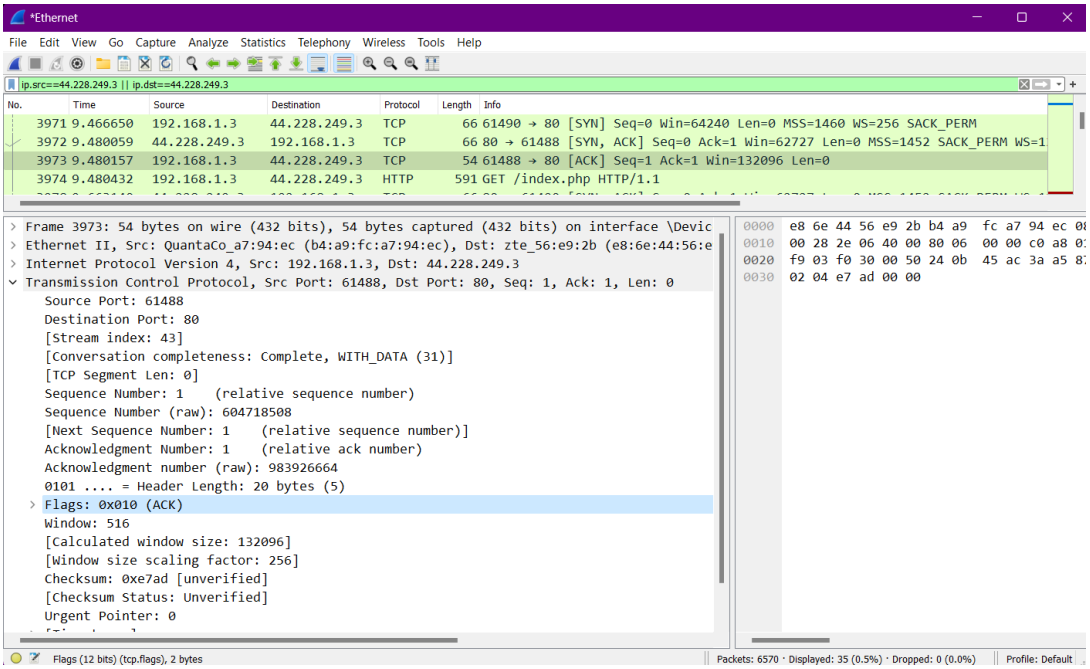
Hình 3. 7: Gói tin TCP chứa SYN flag

Đây là gói tin TCP của client browser gửi tới web server có nội dung thể hiện việc máy client đã gửi một gói tin yêu cầu kết nối có chứa flag Syn từ cổng 61490 tới cổng đích là 80 của web server.

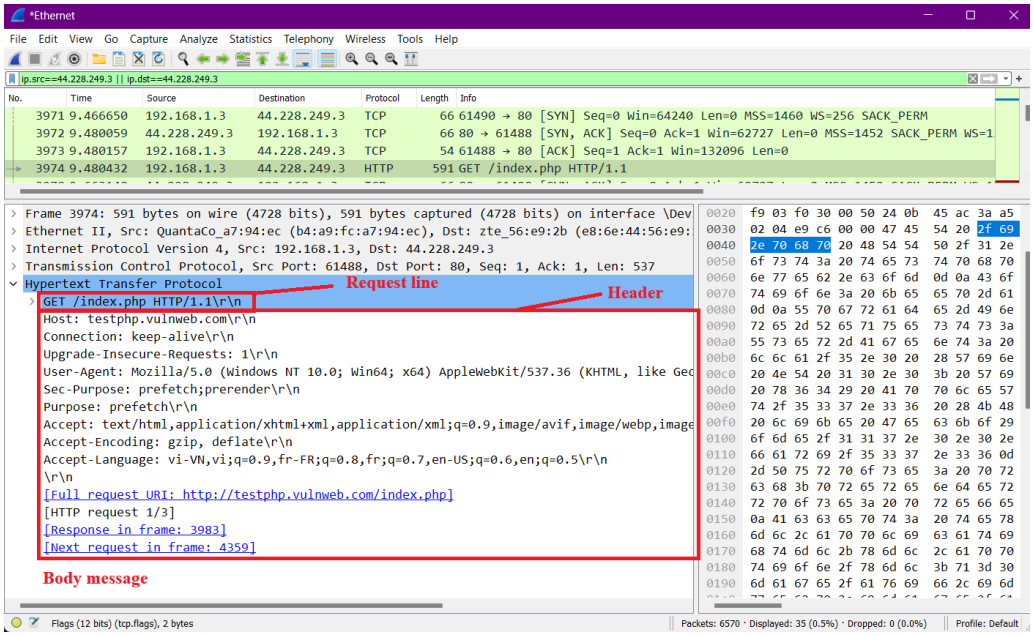


Hình 3. 8: Gói tin TCP chứa SYN, ACK flag

Sau khi nhận được gói tin TCP chứa flag Syn được gửi từ máy client, web server đã phản hồi lại bằng cách gửi lại từ cổng 80 một gói tin chứa flags SYN, ACK tới cổng 61490 của máy client để đồng ý kết nối.

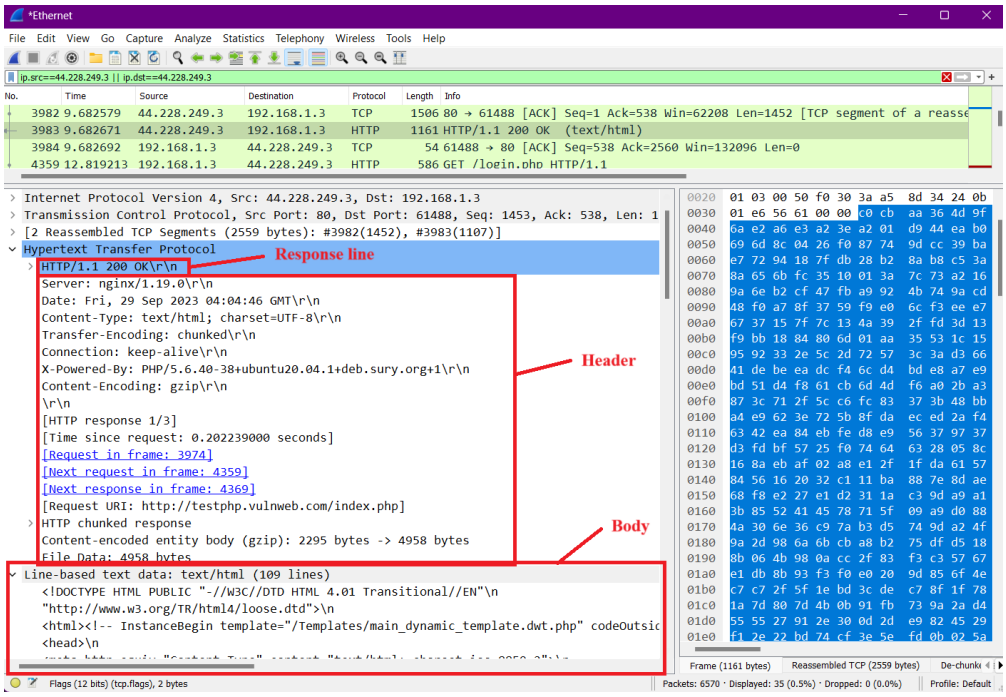


Hình 3. 9: Gói tin TCP chứa ACK flag



Hình 3. 10: Gói tin HTTP Request và các thành phần

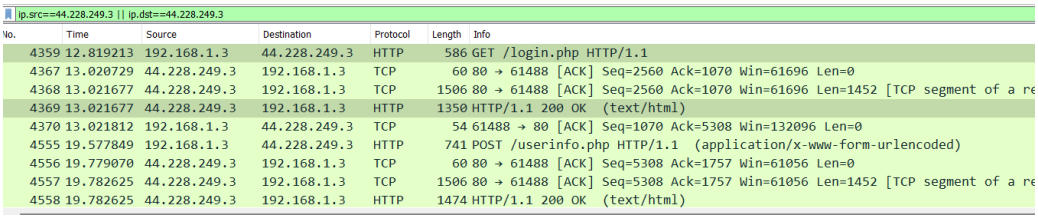
Ngay sau khi gửi gói tin chứa flag Ack, máy client đã đồng thời gửi một gói tin sử dụng giao thức HTTP từ cổng 61490, có chứa HTTP Request trong đó gồm 3 phần. Phần đầu là Request line bao gồm: phương thức GET tới path /index.php và phiên bản giao thức HTTP/1. Phần thứ hai là Header bao gồm các trường giá trị như: host là testphp.vulnweb.com, connection (trạng thái kết nối), user-agent (thông tin máy client)… Và phần cuối cùng là Body message đang trống. Gói tin này được gửi tới cổng 80 của web server để yêu cầu tài nguyên.



Hình 3. 11: Gói tin HTTP Response và các thành phần

Sau khi xác lập kết nối xong, web server đã phản hồi lại gói tin HTTP Request của máy client bằng gói tin HTTP Response gồm 3 thành phần. Phần đầu tiên là Response line có chứa phiên bản của giao thức HTTP là HTTP/1.1 và mã trạng thái là 200. Phần thứ hai là các trường dữ liệu chứa thông tin bổ sung về respose bao gồm: server, date, content-type,… Phần cuối cùng là phần body, đây chính là phần chứa các dữ liệu thực tế của response như nội dung web, tập tin hoặc các thông tin khác.

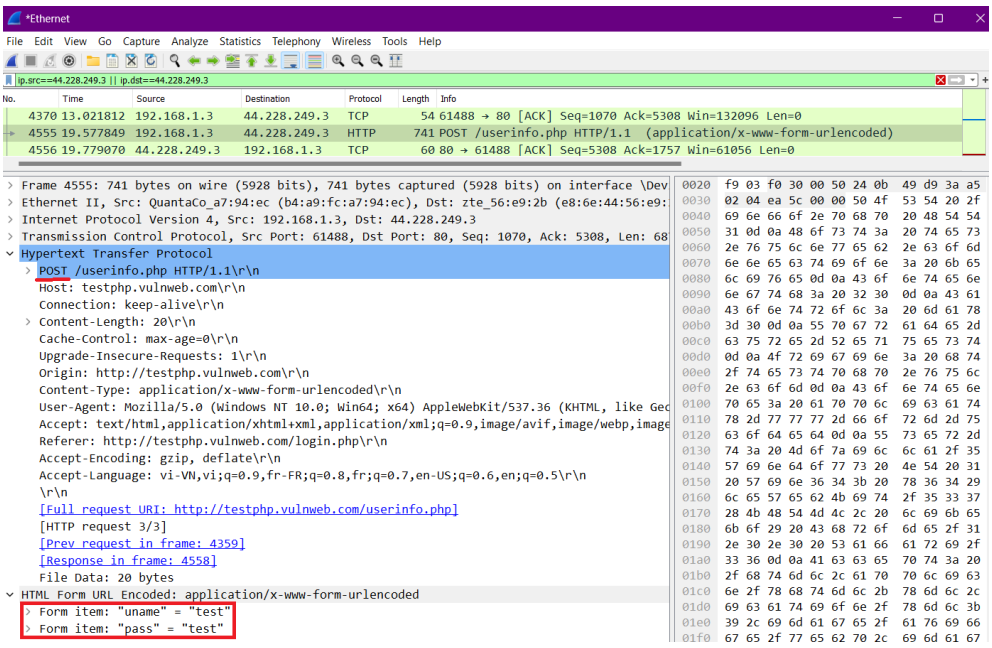
Tiếp theo, ta tiến hành quan sát các gói tin HTTP khi ta thực hiện việc đăng nhập và hệ thống của web server.



Hình 3. 12: Các gói tin thể hiện quá trình đăng nhập vào hệ thống

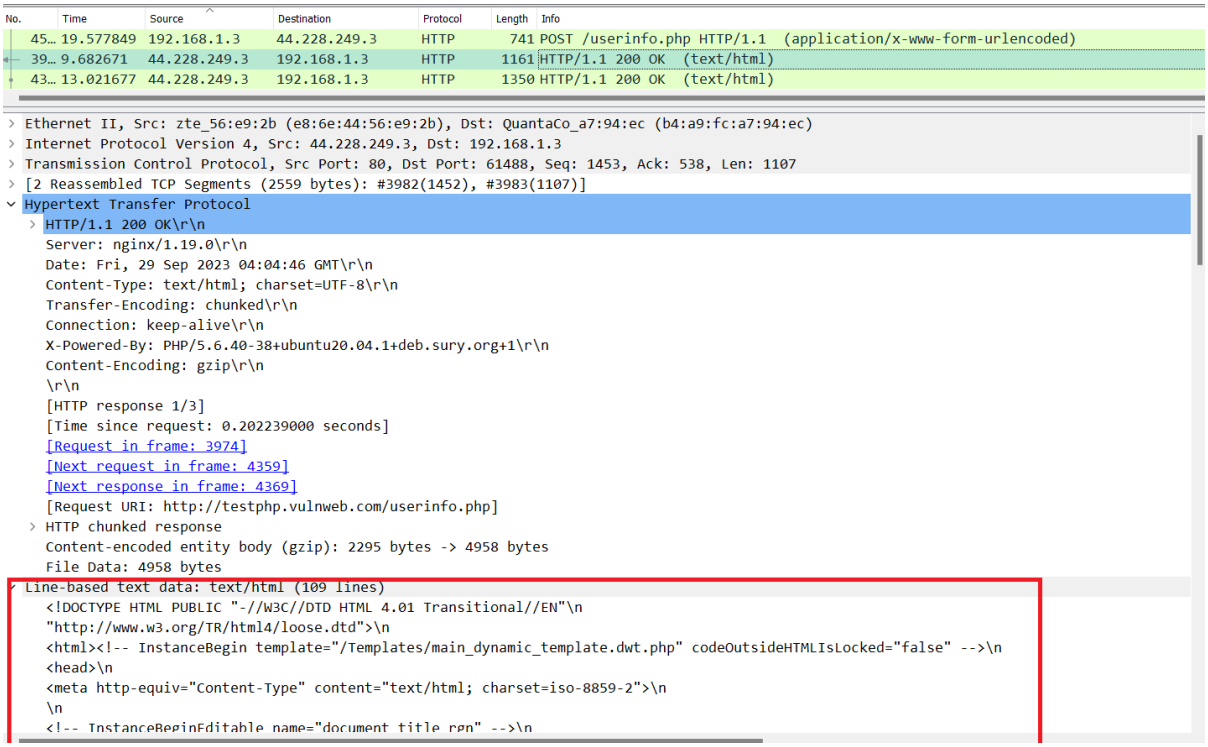
Có thể thấy client browser đã gửi gói tin HTTP Request với thông tin ‘GET /login.php HTTP/1.1’ yêu cầu dữ liệu của đường dẫn /login.php để có thể truy cập tới giao diện đăng nhập của hệ thông. Và web server đã phản hồi lại bằng gói tin HTTP Response với thông tin ‘HTTP/1.1 200 OK’ để thông báo yêu câu đã được thực hiện thành công, kèm theo đó là dữ liệu của đường dẫn /login.php.

Tiến hành phân tích quá trình client gửi dữ liệu để đăng nhập vào hệ thống web server.



Hình 3. 13: Gói tin HTTP Request với phương thức POST

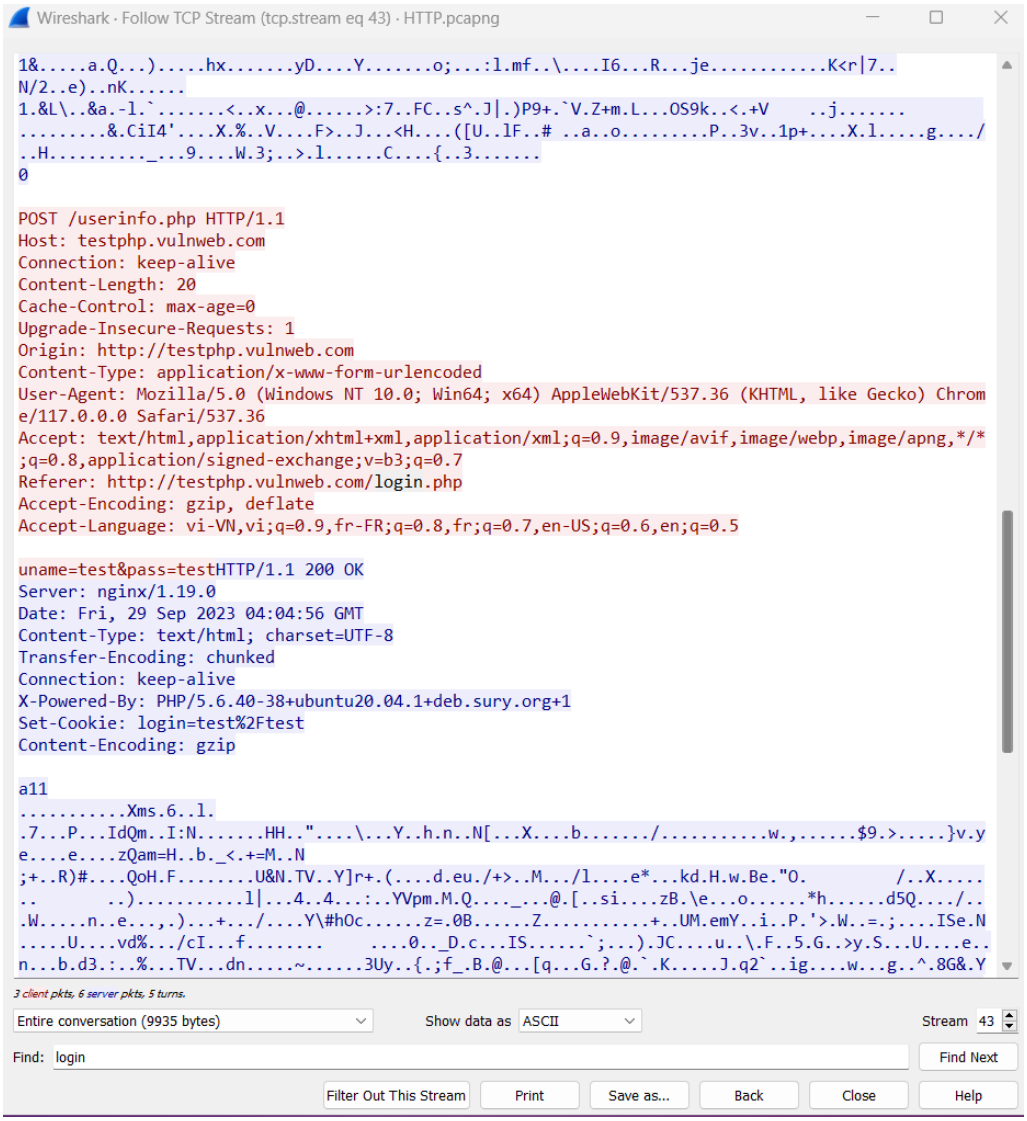
Trong gói tin HTTP Request lần này, client browser gửi HTTP Request với phương thức POST thể hiện việc máy client đang gửi thông tin đăng nhập với web server tới địa chỉ dẫn là /userinfo.php. Kèm theo đó là dữ liệu đăng nhập được lưu ở phần body của request, có nội dung là ‘uname’ có giá trị ‘test’ và ‘pass’ có giá trị ‘test’.



Hình 3. 14: Gói tin HTTP Response

Sau khi xác thực thông tin đăng nhập trong HTTP Request được gửi tới là đúng, web server đã phản hồi lại client bằng HTTP Response thông báo yêu cầu thành công và kèm theo đó là dữ liệu của địa chỉ đường dẫn /userinfo.php chứa thông tin của người đăng nhập.

Tiến hành theo dõi tổng quát các gói tin HTTP, bằng cách sử dụng chức năng Follow HTTP stream, ta có thể thấy các dữ liệu được trao đổi giữa hai máy client và server rõ ràng.



Hình 3. 15: Các dữ liệu trong gói tin HTTP được trao đổi giữa hai máy

3.1.3 Đánh giá

Qua quá trình phân tích các gói tin HTTP, có thể thấy HTTP là một giao thức quan trọng và cơ bản trong việc truyền tải thông tin trên internet. Tuy nhiên, HTTP cũng có nhược điểm về bảo mật là chúng không hề được mã hóa và bảo mật. Đây chính là kẽ hở mà nhiều hacker đã lợi dụng để đánh cắp thông tin người dùng.

3.2 Phân tích gói tin HTTPS

3.2.1 Mục tiêu

Lần này, tiến hành truy cập vào một website sử dụng giao thức HTTPS, và sử dụng công cụ Wireshark để quan sát và phân tích các gói tin giao tiếp giữa client browser và web server. Từ đó có thể hiểu rõ được các chứng chỉ bảo mật cũng như cách hoạt động của giao thức HTTPS.

3.2.2 Thực hiện

- **Bước 1: chuẩn bị trước phân tích**

Kích hoạt wireshark và chọn network interface phù hợp. Sau đó, từ browser, truy cập tới website có host ‘actvn.edu.vn’. Có thể thấy website này đã và đang sử dụng giao thức HTTPS để giao tiếp với các client browser.

Hình 3. 16: Web của ‘actvn.edu.vn’

Sử dụng lệnh ‘nslookup’ trên cmd để tìm được IP của web server. Sau khi xong, ta nhận được ip address là 42.112.213.84.

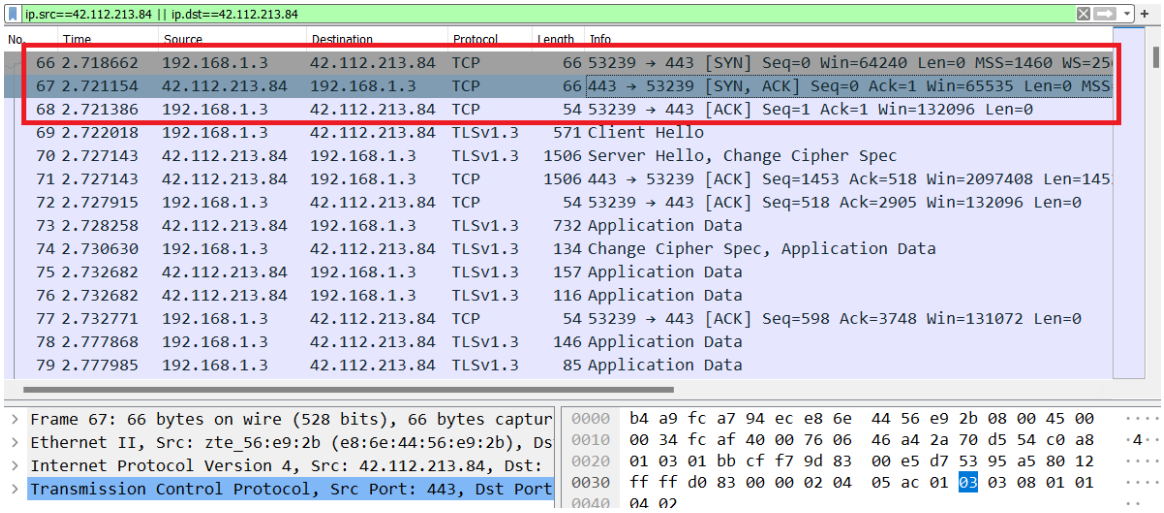
```
┌──(txc3000㉿txc3000)-[~]
└─$ nslookup actvn.edu.vn
Server:         172.18.128.1
Address:        172.18.128.1#53

Non-authoritative answer:
Name:   actvn.edu.vn
Address: 42.112.213.84
```

Hình 3. 17: IP của host ‘actvn.edu.vn’

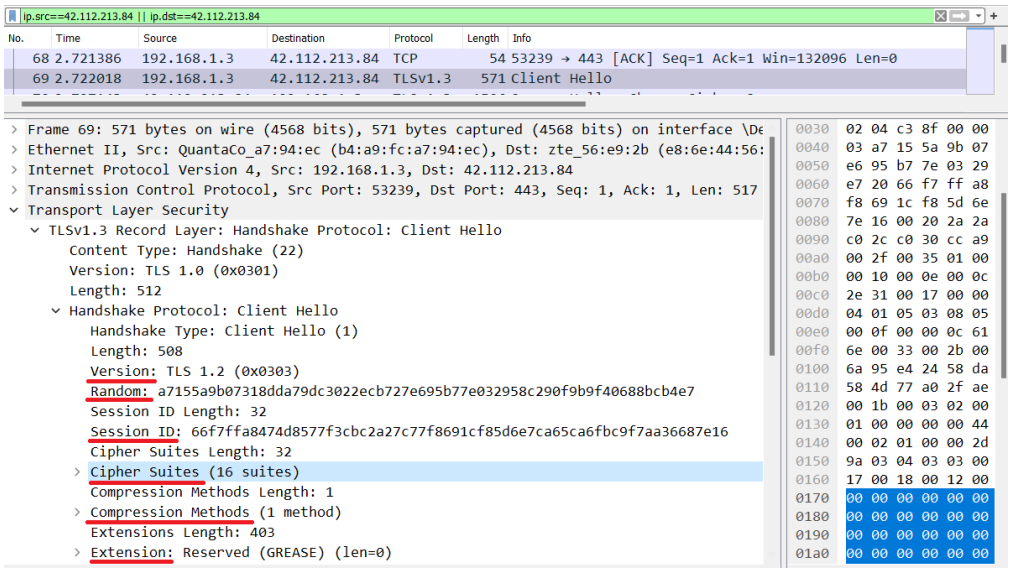
- Bước 2: Phân tích gói tin

Sau khi truy cập tới website của host ‘actvn.edu.vn’, tiến quan sát các gói tin giao tiếp giữa hai máy trên Wireshark. Sử dụng lệnh ‘ip.src==42.112.213.84 || ip.dst==42.112.213.84’ để tiến hành lọc các gói tin có địa chỉ IP nguồn hoặc IP đích là 42.112.213.84.



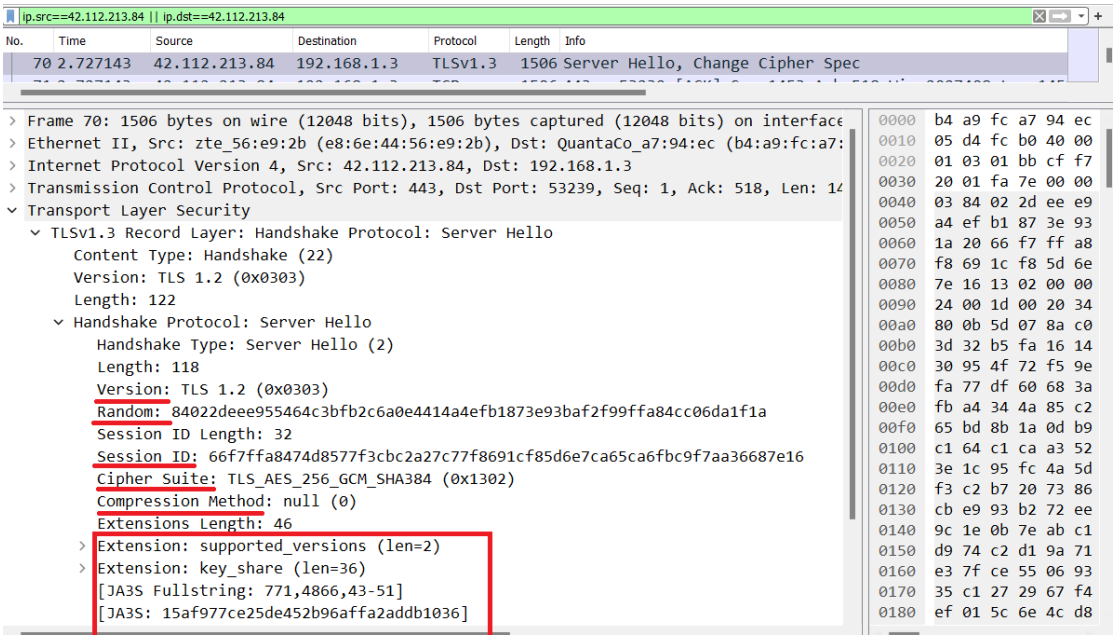
Hình 3. 18: Các gói tin giao tiếp giữa client browser and web server

Như hình trên, trước khi HTTPS được sử dụng, client browser và web server đã thực hiện quá trình 'Bắt tay ba bước' giống như HTTP. Sau khi nhập được gói tin TCP có flag Syn-Ack, client browser sẽ gửi đồng thời 2 gói tin, trong đó có 1 gói tin TCP chứa flag ACK và 1 gói tin TLSv1.3 mở đầu cho quá trình TLS Handshake Protocol có thông tin "Client Hello".

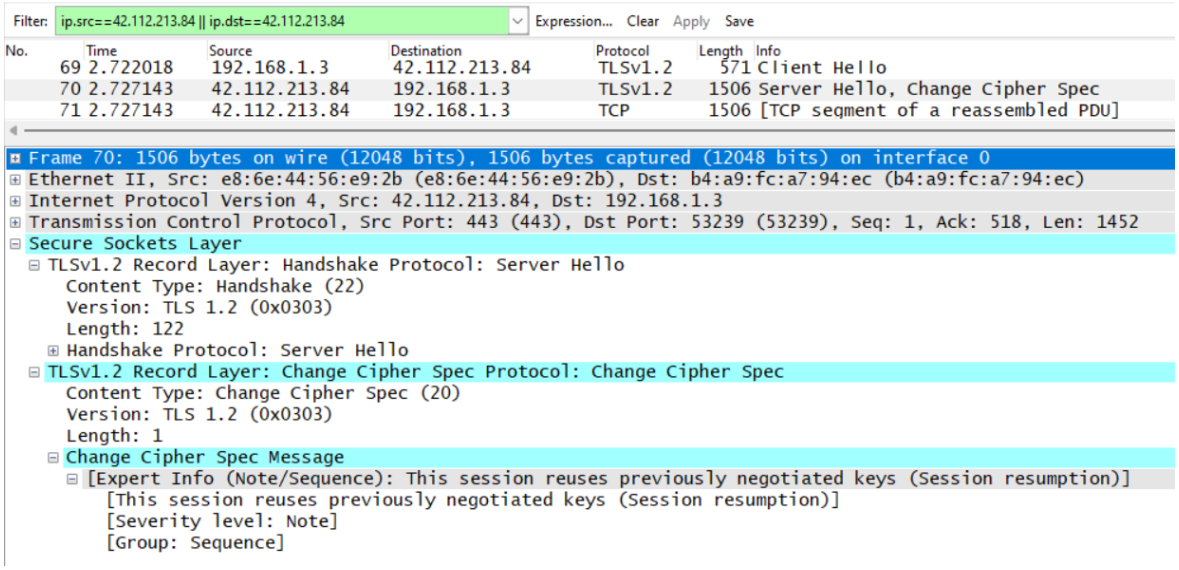


Hình 3. 19: Gói tin Client Hello

Tiến hành phân tích gói tin Client Hello. Gói tin được gửi từ máy client từ cổng 53239 tới cổng 443 của web server. Trong gói tin gồm có 6 phần chính bao gồm: Version: phiên bản TLS, Random: là dãy 32 byte ngẫu nhiên để dùng để đảm bảo cho việc tạo khóa không bị trùng lặp, Session ID: id phiên, Cipher suites: danh sách các loại mã hóa mà máy khách hỗ trợ, Compression method: phương pháp nén dữ liệu mà máy khách hỗ trợ, Extension: các tiện ích mở rộng để có thể cung cấp thêm thông tin.

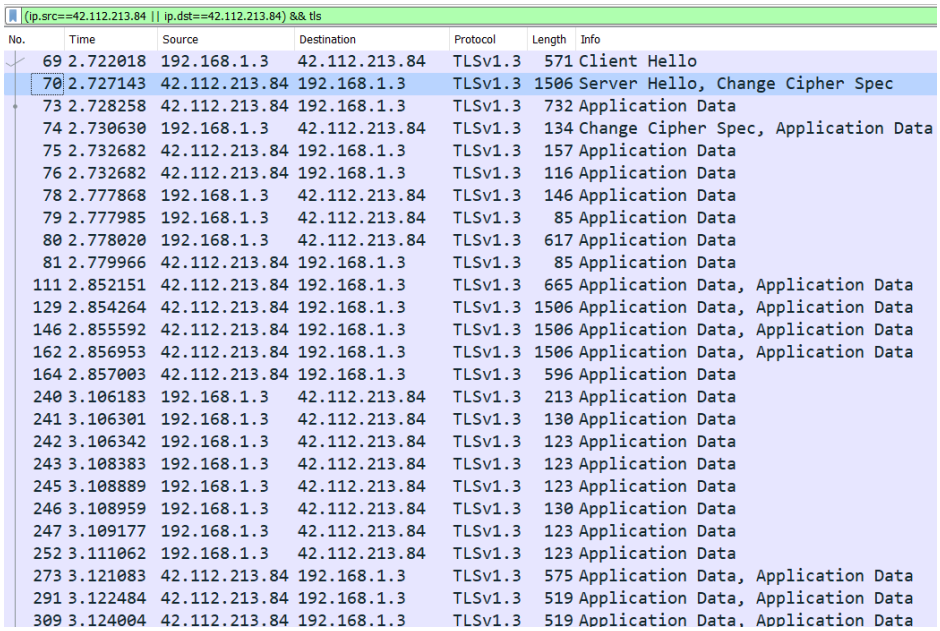


Hình 3. 20: Gói tin Server Hello



Hình 3. 21: Phần Change Cipher Spec trong gói tin

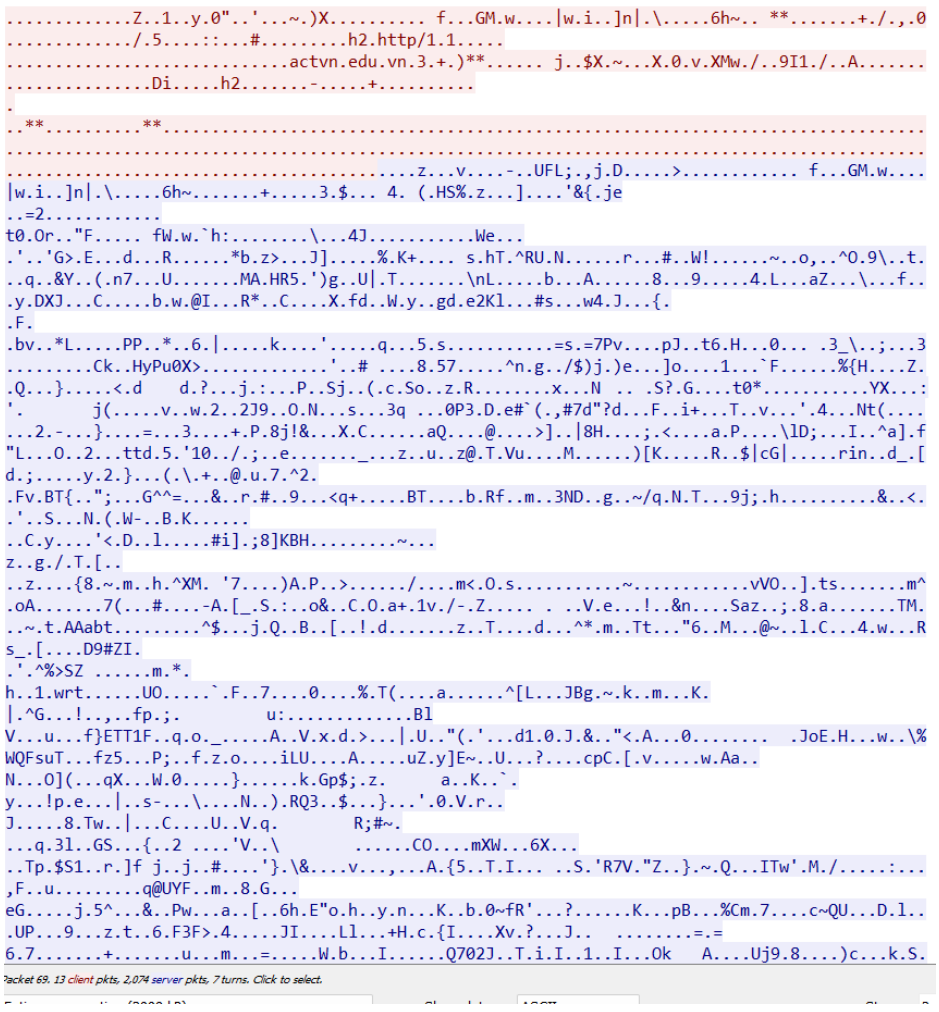
Sau khi web server nhận được gói tin Client Hello được gửi đi từ client browser, server đã phản hồi bằng gói tin Server Hello được gửi từ cổng 443 tới cổng 53239 máy client. Trong gói tin này chứa phiên bản TLS, session id, loại mã hóa sẽ được dùng để mã hóa đường truyền: trong gói tin này sử dụng mã hóa 'TLS_AES_256_GCM_SHA384'. Kèm theo gói tin còn có thêm một phần là Change Cipher Spec, đây là phần quan trọng dùng để thông báo rằng các dữ liệu sẽ được mã hóa bằng mã hóa đã chọn.



(ip.src==42.112.213.84 || ip.dst==42.112.213.84) && tls

No.	Time	Source	Destination	Protocol	Length	Info
69	2.722018	192.168.1.3	42.112.213.84	TLSv1.3	571	Client Hello
70	2.727143	42.112.213.84	192.168.1.3	TLSv1.3	1506	Server Hello, Change Cipher Spec
73	2.728258	42.112.213.84	192.168.1.3	TLSv1.3	732	Application Data
74	2.730630	192.168.1.3	42.112.213.84	TLSv1.3	134	Change Cipher Spec, Application Data
75	2.732682	42.112.213.84	192.168.1.3	TLSv1.3	157	Application Data
76	2.732682	42.112.213.84	192.168.1.3	TLSv1.3	116	Application Data
78	2.777868	192.168.1.3	42.112.213.84	TLSv1.3	146	Application Data
79	2.777985	192.168.1.3	42.112.213.84	TLSv1.3	85	Application Data
80	2.778020	192.168.1.3	42.112.213.84	TLSv1.3	617	Application Data
81	2.779966	42.112.213.84	192.168.1.3	TLSv1.3	85	Application Data
111	2.852151	42.112.213.84	192.168.1.3	TLSv1.3	665	Application Data, Application Data
129	2.854264	42.112.213.84	192.168.1.3	TLSv1.3	1506	Application Data, Application Data
146	2.855592	42.112.213.84	192.168.1.3	TLSv1.3	1506	Application Data, Application Data
162	2.856953	42.112.213.84	192.168.1.3	TLSv1.3	1506	Application Data, Application Data
164	2.857003	42.112.213.84	192.168.1.3	TLSv1.3	596	Application Data
240	3.106183	192.168.1.3	42.112.213.84	TLSv1.3	213	Application Data
241	3.106301	192.168.1.3	42.112.213.84	TLSv1.3	130	Application Data
242	3.106342	192.168.1.3	42.112.213.84	TLSv1.3	123	Application Data
243	3.108383	192.168.1.3	42.112.213.84	TLSv1.3	123	Application Data
245	3.108889	192.168.1.3	42.112.213.84	TLSv1.3	123	Application Data
246	3.108959	192.168.1.3	42.112.213.84	TLSv1.3	130	Application Data
247	3.109177	192.168.1.3	42.112.213.84	TLSv1.3	123	Application Data
252	3.111062	192.168.1.3	42.112.213.84	TLSv1.3	123	Application Data
273	3.121083	42.112.213.84	192.168.1.3	TLSv1.3	575	Application Data, Application Data
291	3.122484	42.112.213.84	192.168.1.3	TLSv1.3	519	Application Data, Application Data
309	3.124004	42.112.213.84	192.168.1.3	TLSv1.3	519	Application Data, Application Data

Hình 3. 22: Các gói tin TLS giữa client và server



```
.............Z..1..y.0"..'...~.)X.......... f...GM.w....|w.i..]n|.\.....6h~.. **.......+./.,.0
............./.5...::...#.........h2.http/1.1.....
.............................actvn.edu.vn.3.+.)**...... j..$X.~...X.0.v.XMw./..9I1./..A.......
..............Di.....h2.......-.....+..........
.
..**..........**.....................................................................................
.....................................................................................................
.....................................................................................................
.....................................z...v....-..UFL;.,j.D.....>.............. f...GM.w....
|w.i..]n|.\.....6h~.......+.....3.$... 4. (.HS%.z...]....'&{.je
..=2.............
t0.Or.."F..... fW.w.`h:........\...4J...........We...
.'..'G>.E...d...R......*b.z>...J].....%.K+.... s.hT.^RU.N......r...#..W!......~..o,..^O.9\..t.
..q..&Y..(.n7...U.......MA.HR5.')g..U|.T.......\nL.....b...A......8...9.....4.L...aZ...\...f..
.y.DXJ...C.....b.w.@I...R*..C....X.fd..W.y..gd.e2Kl...#s...w4.J...{.
.F.
.bv..*L.....PP..*..6.|.....k....'.....q...5.s...........=s.=7Pv....pJ..t6.H...0... .3_\..;...3
.........Ck..HyPu0X>.............'..# ....8.57.....^n.g../$)j.)e...]o....1...`F......%{H....Z.
.Q...}.....<.d    d.?...j.:...P..Sj..(.c.So..z.R........x...N  .. .S?.G....t0*...........YX...:
'.       j(.....v..w.2..2J9..O.N...s...3q ...0P3.D.e#`(.,#7d"?d...F..i+...T..v...'.4...Nt(....
...2.-...}....=...3....+.P.8j!&...X.C......aQ....@....>]..|8H....;.<....a.P....\lD;...I..^a].f
"L...O..2...ttd.5.'10../.;..e......._...z..u..z@.T.Vu....M......)[K.....R..$|cG|.....rin..d_.[
d.;.....y.2.}...(.\.+..@.u.7.^2.
.Fv.BT{..";...G^^=...&..r.#..9...<q+.....BT....b.Rf..m..3ND..g..~/q.N.T...9j;.h..........&..<.
.'..S...N.(.W-..B.K......
..C.y....'<.D..l.....#i].;8]KBH.........~...
z..g./.T.[..
..z....{8.~.m..h.^XM. '7....)A.P..>....../....m<.O.s...........~............vVO..].ts.......m^
.oA.......7(...#....-A.[_.S.:..o&..C.O.a+.1v./-.Z..... . ..V.e...!..&n....Saz..;.8.a.......TM.
..~.t.AAabt.........^$...j.Q..B..[..!.d.......z..T....d...^*.m..Tt..."6..M...@~..l.C...4.w...R
s_.[....D9#ZI.
.'.^%>SZ ......m.*.
h..1.wrt......UO.....`.F..7....0....%.T(....a......^[L...JBg.~.k..m...K.
|.^G...!..,..fp.j.         u:.............Bl
V...u...f}ETT1F..q.o._.....A..V.x.d.>...|.U.."(.'...d1.0.J.&.."<.A...0........  .JoE.H...w..\%
WQFsuT...fz5...P;..f.z.o....iLU....A.....uZ.y]E~..U...?....cpC.[.v.....w.Aa..
N...O](...qX...W.0.....}......k.Gp$;.z.      a..K..`.
y...!p.e...|..s-...\....N..).RQ3..$...}...'.0.V.r..
J.....8.Tw..|...C....U..V.q.        R;#~.
...q.3l..GS...{..2 ....'V..\        ......CO....mXW...6X...
..Tp.$S1..r.]f j..j..#....'}.\&....v...,...A.{5..T.I... ..S.'R7V."Z..}.~.Q...ITw'.M./.....:...
,F..u.........q@UYF..m..8.G...
eG.....j.5^...&..Pw...a..[..6h.E"o.h..y.n...K..b.0~fR'...?......K...pB...%Cm.7....c~QU...D.l..
.UP...9...z.t..6.F3F>.4.....JI....Ll...+H.c.{I....Xv.?...J..  ........=.=
6.7.......+.......u...m...=.....W.b...I......Q702J..T.i.I..1..I...Ok   A....Uj9.8....)c...k.S.
```

Packet 69. 13 client pkts, 2,074 server pkts, 7 turns. Click to select.

Hình 3. 23: Các gói tin đã được mã hóa

Sau khi nhận gói tin Server Hello, máy client và server đã xác nhận được loại thông tin mã hóa và các thông tin bổ sung. Từ đó 2 máy đã giao tiếp với nhau bằng cách trao đổi các gói tin dữ liệu đã được mã hòa bằng loại mã hóa được chọn trong gói tin Server Hello và kết quả là các ký tự mà chúng ta không thể đọc được như hình trên.

3.2.3 Đánh giá

Từ kết quả thu được từ quá trình phân tích trên, có thể thấy HTTPS là phiên bản bảo mật của giao thức HTTP. Nó giúp việc đảm bảo an toàn, bảo mật và toàn vẹn trên đường truyền dữ liệu bằng cách các gói tin từ máy gửi sẽ mã hóa trước khi truyền đi, và các dữ liệu sẽ được mã hóa trong suốt toàn bộ đường truyền, khi tới máy nhận, các gói dữ liệu sẽ được giải mã.

KẾT LUẬN

Qua quá trình nghiên cứu về giao thức HTTP (Hypertext Transfer Protocol) và HTTPS (Hypertext Transfer Protocol Secure), đã giúp ta nhận thấy sự quan trọng hai giao thức này mang lại trong cuộc sống và công nghệ hiện đại. Chúng không chỉ đơn giản là các kỹ thuật kết nối mạng, mà còn là nền tảng cơ bản cho mọi trải nghiệm trực tuyến của chúng ta.

Giao thức HTTP có hiệu suất là yếu tố không thể bỏ qua đối với trải nghiệm người dùng trực tuyến. Cải tiến hiệu suất và tối ưu hóa cách hoạt động của HTTP là nhiệm vụ quan trọng để đảm bảo tốc độ và ổn định trong truy cập web.

Giao thức HTTPS đóng vai trò quan trọng trong việc bảo vệ dữ liệu và thông tin cá nhân của người dùng trên internet. Ngoài ra chúng còn đảm bảo rằng chúng ta có thể tương tác an toàn trên internet. Tuy nhiên, việc duy trì mức độ an toàn này đòi hỏi sự cải tiến liên tục để đối phó với các mối đe dọa ngày càng phức tạp.

TÀI LIỆU THAM KHẢO

- https://www.w3.org/Protocols/rfc2616/rfc2616.html
- https://developer.mozilla.org/vi/docs/Web/HTTP
- https://vi.wikipedia.org/wiki/Hypertext_Transfer_Protocol
- https://vi.wikipedia.org/wiki/HTTPS
- https://cystack.net/blog/http-va-https-la-gi
- https://medium.com/devops-world/http-https-analysis-using-wireshark-cbe07c23520
- https://www.youtube.com/watch?v=j9QmMEWmcfo&t=197s&ab_channel=ByteByteGo
- https://www.wireshark.org/

PHỤ LỤC

Wireshark: phần mềm mạng được sử dụng để phân tích và kiểm tra gói dữ liệu trên mạng. Nó cho phép bạn theo dõi và ghi lại các gói dữ liệu đi qua mạng, từ đó bạn có thể phân tích chúng để hiểu về hoạt động mạng, xác định vấn đề và tìm kiếm lỗ hổng bảo mật.

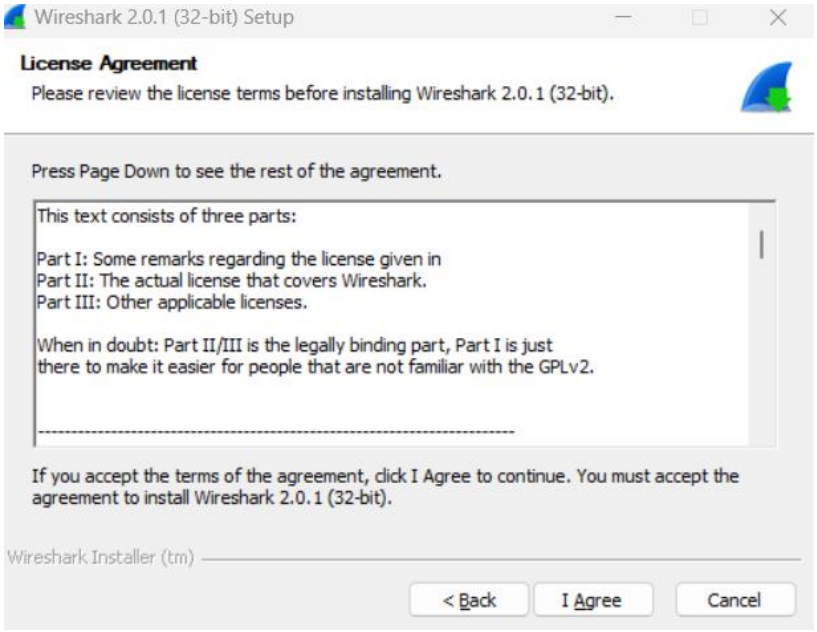
- **Cài đặt công cụ:**

Bước 1: Tiến hành tải xuống công cụ bằng đường link dưới đây:

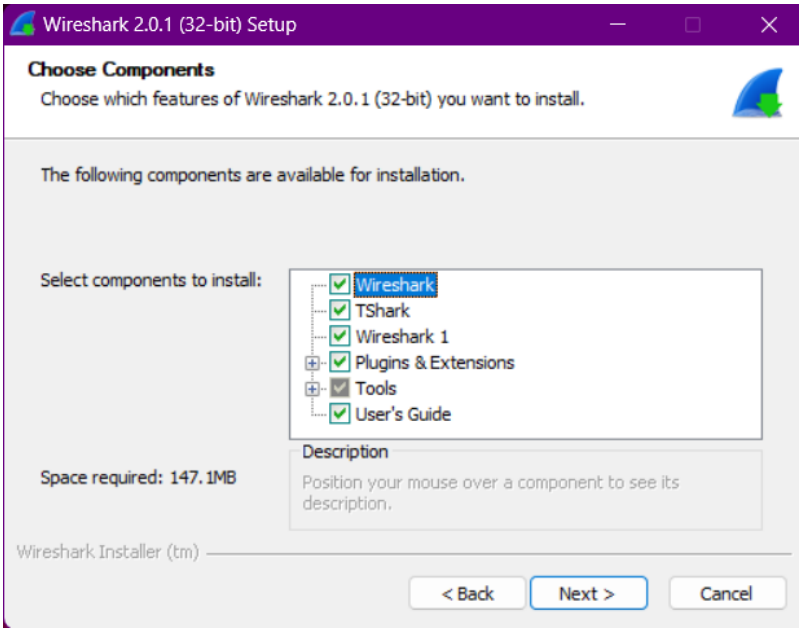
https://2.na.dl.wireshark.org/win64/Wireshark-win64-4.0.10.exe

Bước 2: Sau khi tải về xong, tiến hành click đúp vào file để tiến hành cài đặt.

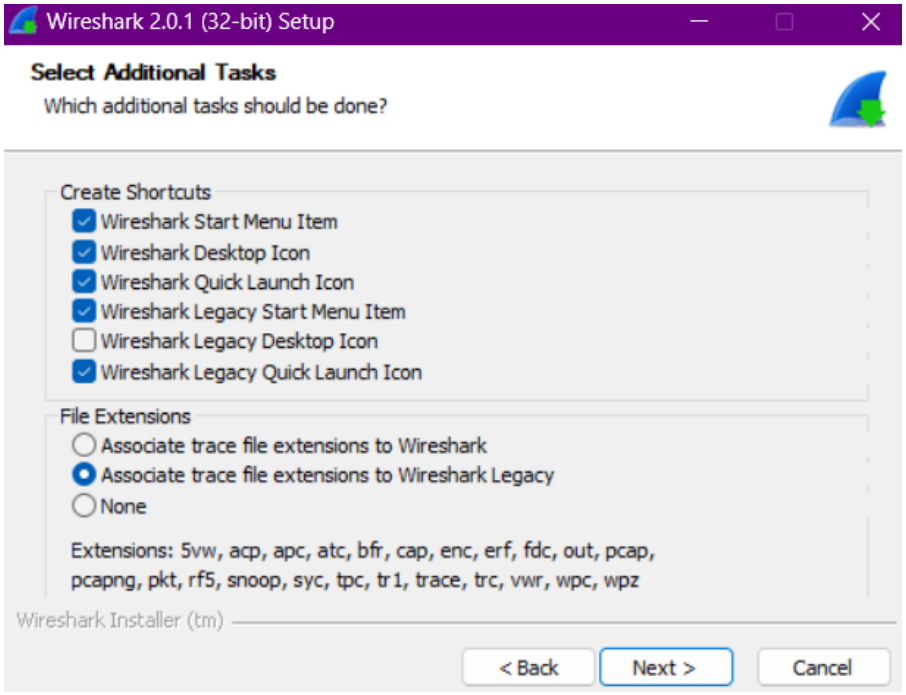
Bước 3: Click nút "Next" và tiếp theo chọn "I Agree"



Bước 4: Xem và cài thêm các thành phần phụ, sau đó chọn "Next" để tiếp tục cài đặt.



Bước 5: Chọn thêm các Task và "Next" để tiếp tục.



Bước 6: Tiếp tục chọn "Next" 2 lần và chọn "Install" để tiến hành cài đặt.

Bước 7: Sau khi cài xong chọn"Next" và "Finish" để hoàn tất.