

## **Progress Report 4**

### **Kelompok 2**

Dave Travis - 2201020008  
Muhammad Noval - 2201020014  
Rizsky Parsadanta R. - 2201020117  
Arya Winata - 2201020001

Judul proyek : Analisis Serangan Sniffing & Mitigasinya di Wireshark

Tujuan proyek : Memahami cara kerja penyerangan *packet sniffing* dan metode pencegahannya.

Target proyek : Mengaktifkan HTTPS & HSTS  
(Minggu 4)

### Pengantar:

Pada progress minggu keempat ini, kami akan melakukan hal yang cukup serupa dengan apa yang kami lakukan minggu lalu, yaitu mencoba menganalisa paket traffic yang terjadi ketika client hendak melakukan login web server, hanya saja minggu lalu kami fokus ke protokol HTTP, yang mana terbukti tidak aman, dan kali ini kami akan mencoba menggunakan protokol HTTPS yang disertai HSTS.

Dan sedikit info, HTTPS merupakan protokol serupa dengan HTTP, namun lebih aman dikarenakan hasil traffic akan menunjukkan bahwa *password* tidak bisa dibaca oleh penyerang, meski ia berhasil menangkap paketnya. HSTS sendiri merupakan (HTTP Strict Transport Security) adalah mekanisme keamanan web yang memaksa browser untuk selalu menggunakan HTTPS saat berkomunikasi dengan sebuah situs. Untuk percobaan pada minggu ini agak sedikit tidak relevan, karena fokus kita itu analisis *traffic*, namun akan tetap kita coba implementasikan.

Progress ini tetap menggunakan vm di VirtualBox sesuai dengan pelaksanaan dua minggu sebelumnya.

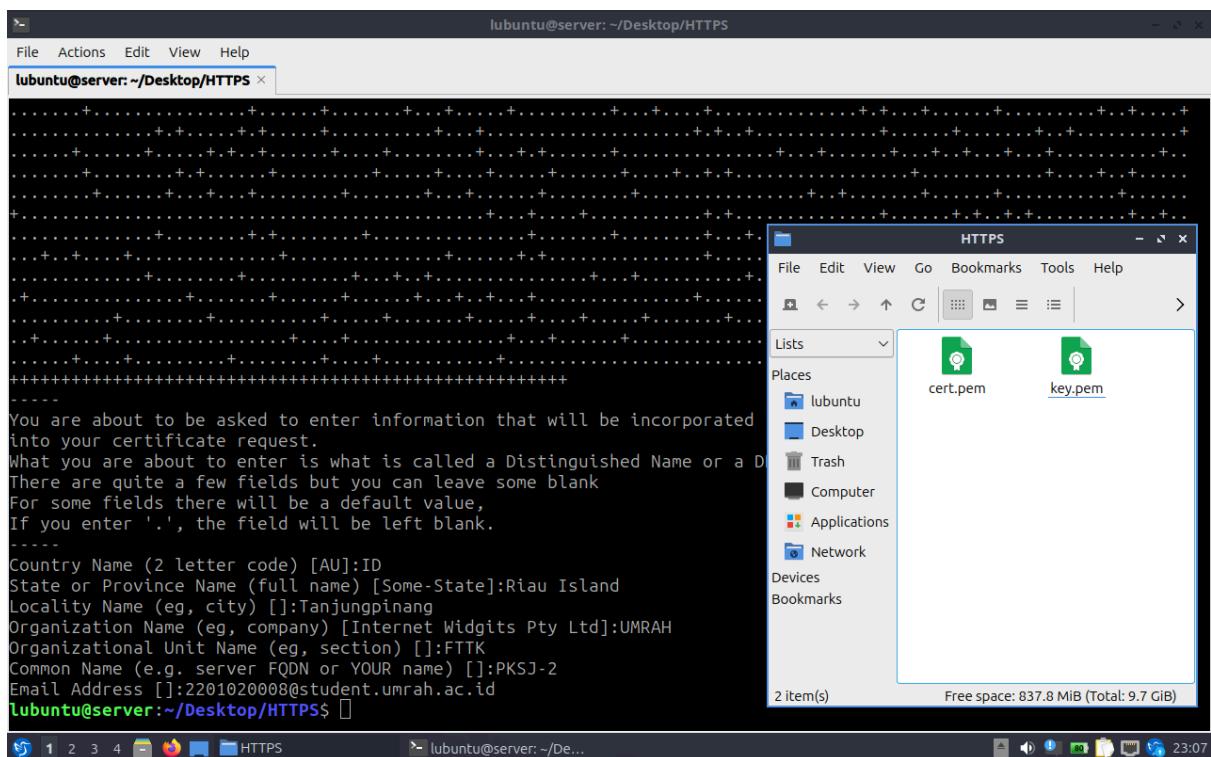
### Pelaksanaan via VirtualBox:

VM yang sudah disiapkan berdasarkan progress minggu lalu ialah 3 vm, yaitu klien, router, dan server. Karena saat ini targetnya ialah analisis traffic HTTPS, sehingga diperlukannya hosting website login yang memakai server address HTTPS, disertai sertifikat SSL yang *self-signed* di vm server selaku host nya. Sehingga sebelum mulai testing, kita akan membuat sertifikat SSL *self-signed* terlebih dahulu, dan mengedit script HTTPS server dari script HTTP minggu lalu.

Sama seperti minggu lalu, kami akan membagi ketiga tugas tersebut terhadap masing-masing vm, di mana hosting website login sederhana akan dilakukan oleh vm server, penguji website login oleh vm klien, dan analisis *traffic* via Wireshark oleh vm router (dikarenakan berada di tengah-tengah topologi, penghubung antara vm klien dan vm server).

### Percobaan:

Tahap pertama ialah kita akan membuat sertifikat SSL *self-signed* di vm server. Kami akan membuat folder bernama HTTPS di Desktop sehingga percobaan kali ini akan dilakukan di folder tersebut.

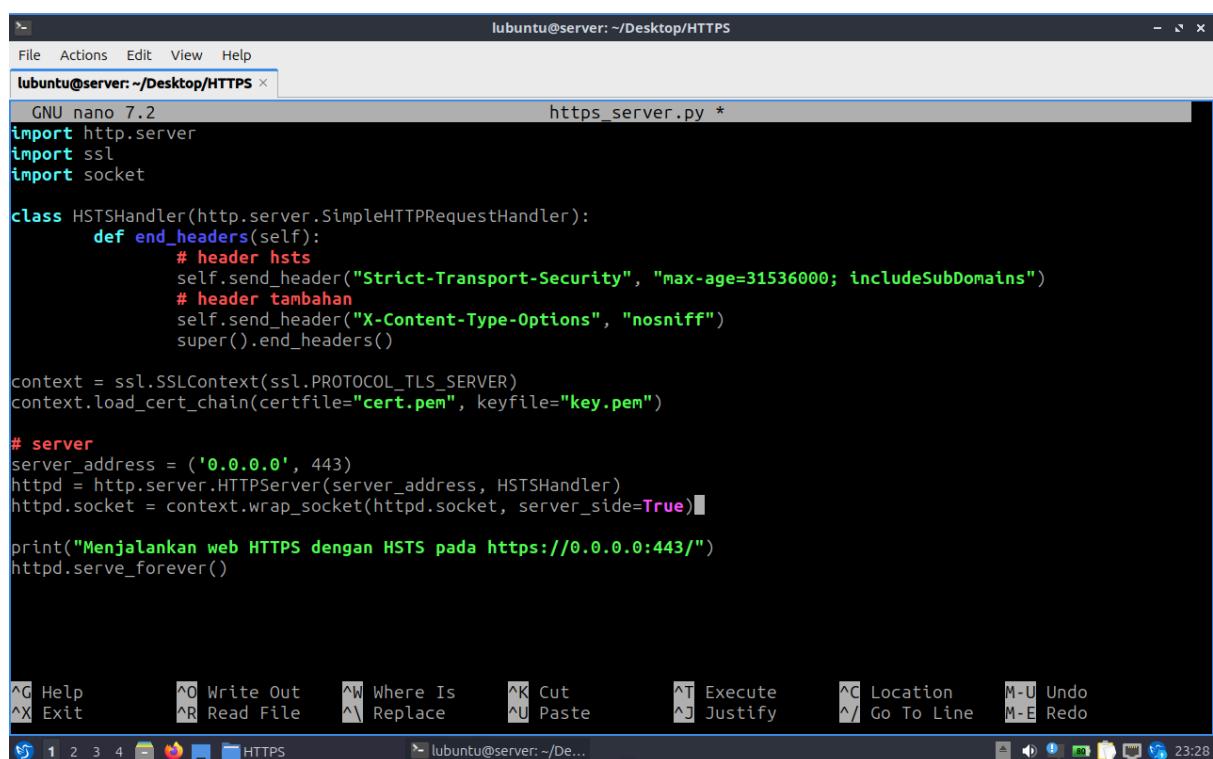


Setelah sertifikat siap dibuat sesuai dengan kredensial kita, maka file *cert.pem* dan *key.pem*, yang menandakan sertifikat berhasil disimpan di folder HTTPS.

Sekarang, kami akan melanjutkan membuat script Python HTTPS pada folder yang sama, sehingga sertifikat SSL dapat dikenali oleh script tersebut. Oh ya, di script ini akan kami asosiasikan dengan HSTS.

Berikut merupakan kode yang akan kami simpan pada folder HTTPS selaku script python untuk server HTTPS:

```
1 import http.server
2 import ssl
3 import socket
4
5 class HSTSHandler(http.server.SimpleHTTPRequestHandler):
6     def end_headers(self):
7         self.send_header("Strict-Transport-Security", "max-age=31536000; includeSubDomains")
8         self.send_header("X-Content-Type-Options", "nosniff")
9         super().end_headers()
10
11 # Buat SSL context (cara modern)
12 context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
13 context.load_cert_chain(certfile="cert.pem", keyfile="key.pem")
14
15 # Bind ke socket
16 server_address = ('0.0.0.0', 443)
17 httpd = http.server.HTTPServer(server_address, HSTSHandler)
18
19 # Wrap socket dengan SSL context
20 httpd.socket = context.wrap_socket(httpd.socket, server_side=True)
21
22 print("Serving HTTPS with HSTS on https://0.0.0.0:443/")
23 httpd.serve_forever()
```



The screenshot shows a terminal window titled 'lubuntu@server: ~/Desktop/HTTPS'. The window contains a nano text editor displaying the Python script for an HTTPS server. The script includes imports for http.server, ssl, and socket, defines a HSTSHandler class that inherits from SimpleHTTPRequestHandler, and sets up an SSL context to wrap the socket. It also prints a message to the console indicating it's serving HTTPS with HSTS on port 443. The terminal window has a standard Linux desktop interface at the bottom, showing icons for various applications like a browser and file manager.

```
lubuntu@server: ~/Desktop/HTTPS
File Actions Edit View Help
lubuntu@server: ~/Desktop/HTTPS ×
GNU nano 7.2                               https_server.py *
import http.server
import ssl
import socket

class HSTSHandler(http.server.SimpleHTTPRequestHandler):
    def end_headers(self):
        # header hsts
        self.send_header("Strict-Transport-Security", "max-age=31536000; includeSubDomains")
        # header tambahan
        self.send_header("X-Content-Type-Options", "nosniff")
        super().end_headers()

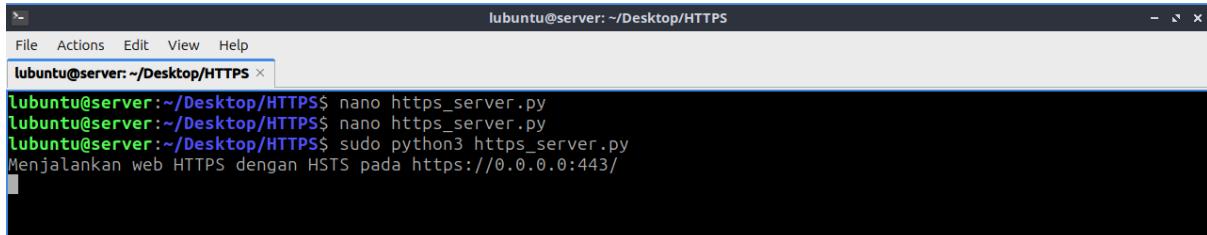
context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
context.load_cert_chain(certfile="cert.pem", keyfile="key.pem")

# server
server_address = ('0.0.0.0', 443)
httpd = http.server.HTTPServer(server_address, HSTSHandler)
httpd.socket = context.wrap_socket(httpd.socket, server_side=True)

print("Menjalankan web HTTPS dengan HSTS pada https://0.0.0.0:443/")
httpd.serve_forever()

^G Help      ^O Write Out   ^W Where Is   ^K Cut       ^T Execute   ^C Location   M-U Undo
^X Exit      ^R Read File   ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line M-E Redo
^Q 1 2 3 4   HTTPS
```

Setelah siap dibuat, maka tinggal kita jalankan saja script servernya.

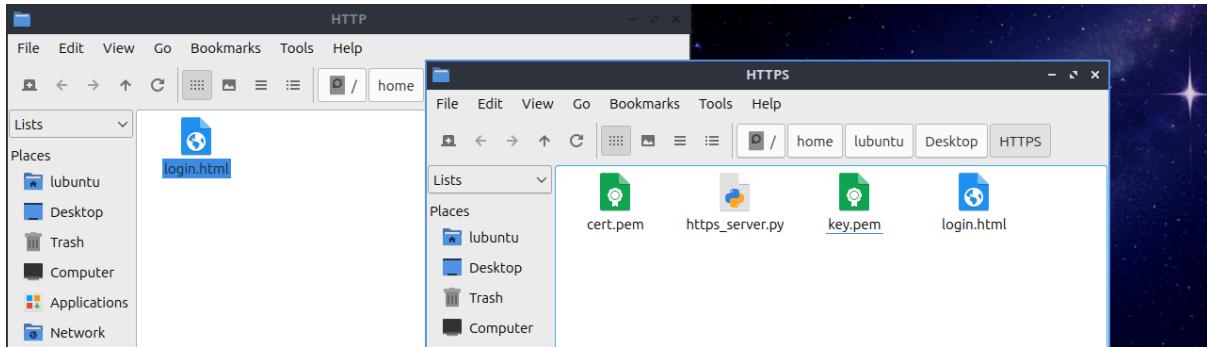


```

lubuntu@server: ~/Desktop/HTTPS
File Actions Edit View Help
lubuntu@server: ~/Desktop/HTTPS ×
lubuntu@server:~/Desktop/HTTPS$ nano https_server.py
lubuntu@server:~/Desktop/HTTPS$ nano https_server.py
lubuntu@server:~/Desktop/HTTPS$ sudo python3 https_server.py
Menjalankan web HTTPS dengan HSTS pada https://0.0.0.0:443/

```

Nah, sudah terlihat kalau web server sudah dijalankan. Dan langkah terakhir ialah memastikan file index.html sebagai percobaan login sudah ada di folder HTTPS. Karena minggu lalu kita buat di folder HTTP, maka kami tinggal menyalin file html tersebut ke folder HTTPS.



Baik, sepertinya sudah semua yang perlu dilakukan di vm server, sekarang saatnya kita boot up vm router dan vm klien, di mana vm klien akan kita coba login pada file yang sama, dan di vm router akan kita pantau dari Wireshark.

Pertama-tama, kita akan coba dari vm klien dulu untuk kita cek apakah web servernya bisa kita akses.



### Warning: Potential Security Risk Ahead

Firefox detected a potential security threat and did not continue to 192.168.2.22. If you visit this site, attackers could try to steal information like your passwords, emails, or credit card details.

[Learn more...](#)

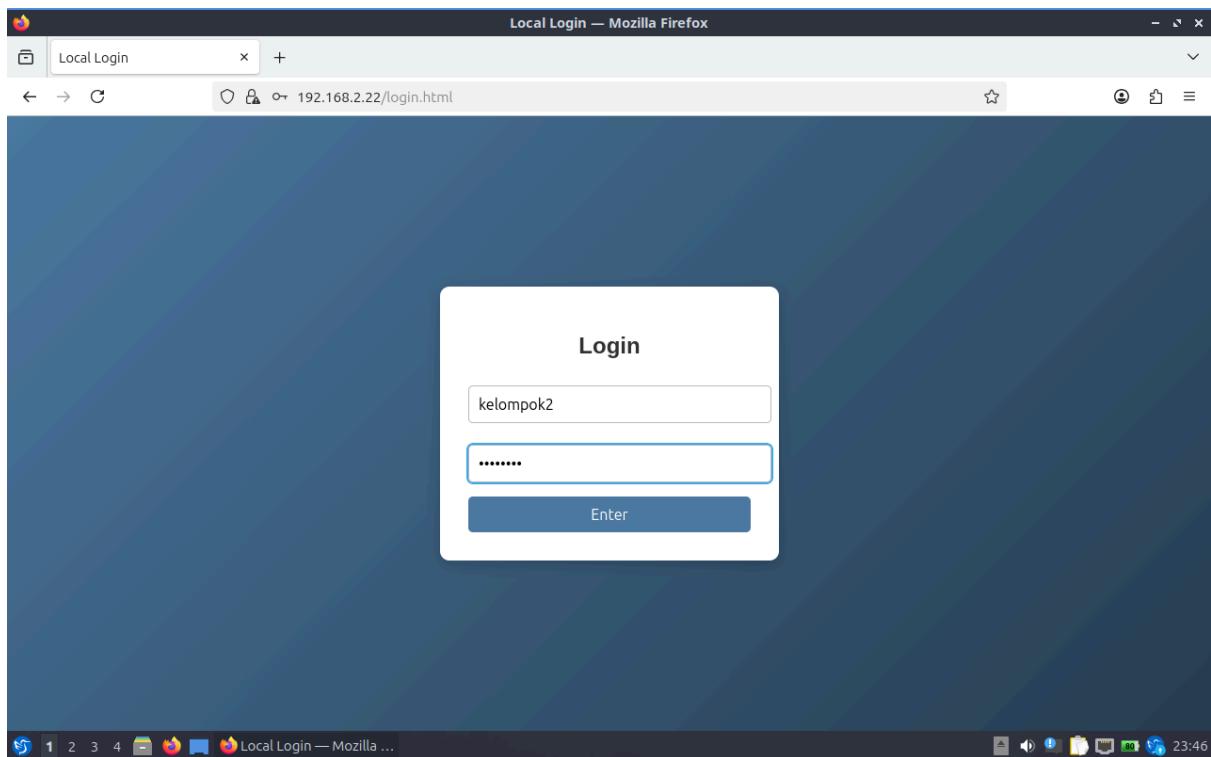
[Go Back \(Recommended\)](#)

[Advanced...](#)

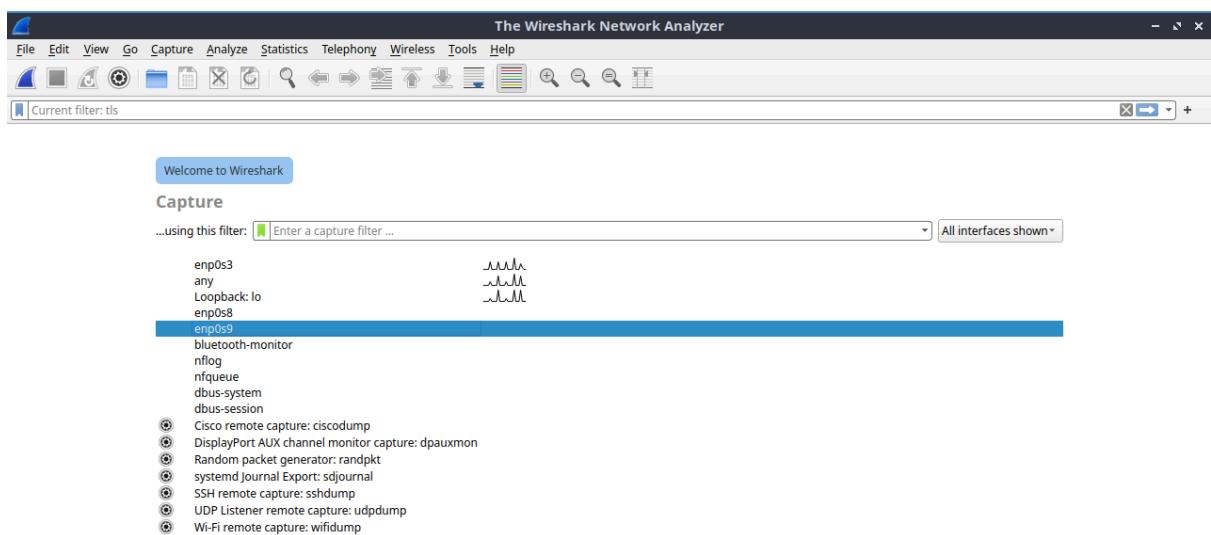
Oops, ada peringatan keamanan, yang menandakan sertifikat SSL kita berjalan dengan baik, jadi tinggal kita setujui saja.

The screenshot shows two tabs in Mozilla Firefox. The first tab, titled "Directory listing for / — Mozilla Firefox", displays a file list with items: cert.pem, https\_server.py, key.pem, and login.html. The second tab, titled "Local Login — Mozilla Firefox", shows a "Login" form with fields for "Username" and "Password" and a "Enter" button. Below the browser window is the Firefox developer tools Network panel. It lists two requests: a GET request for "login.html" (status 200) and a GET request for "favicon.ico" (status 404). The Network panel also shows response headers for "login.html", including "Date: Wed, 03 Dec 2025 16:44:43 GMT", "Server: SimpleHTTP/0.6 Python/3.12.3", "Strict-Transport-Security: max-age=31536000; includeSubDomains", and "X-Content-Type-Options: nosniff".

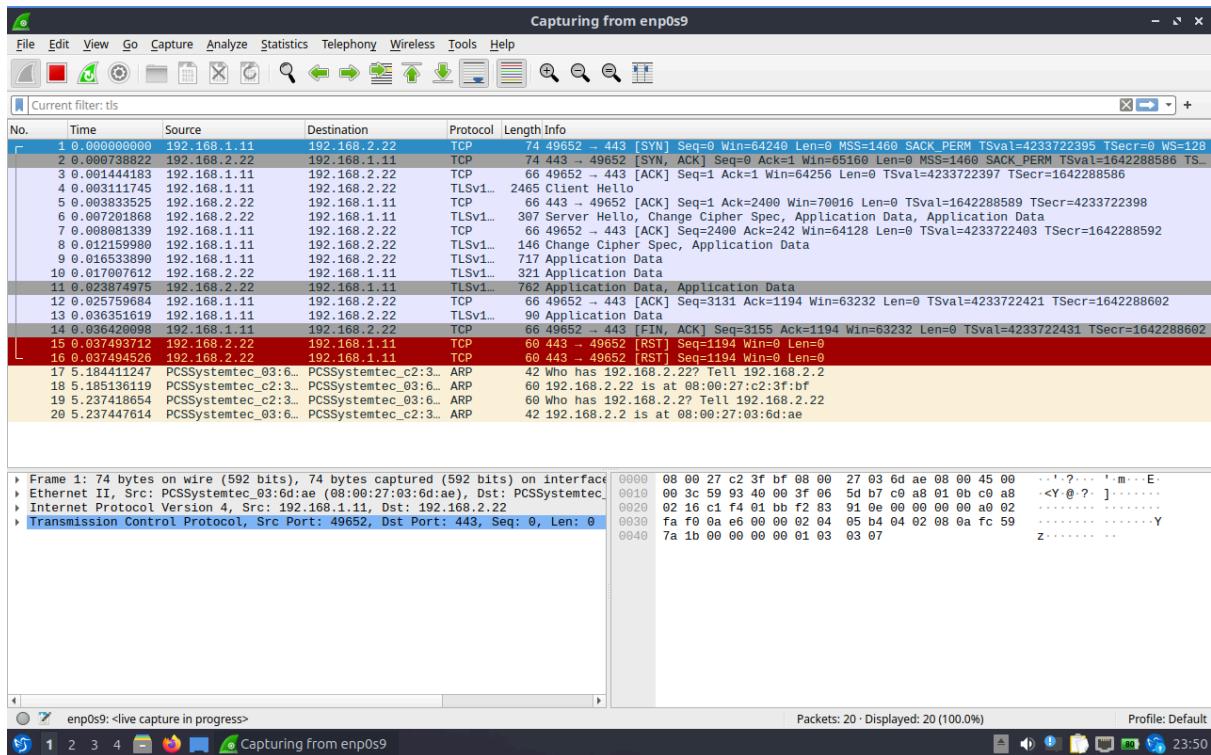
Oke, sudah terlihat halaman login bisa diakses, dan jika kita lihat juga pada inspect network header, pada response headers nya terlihat kalau konfigurasi HSTS kita sudah berjalan. Dan kita lanjutkan proses login kita.



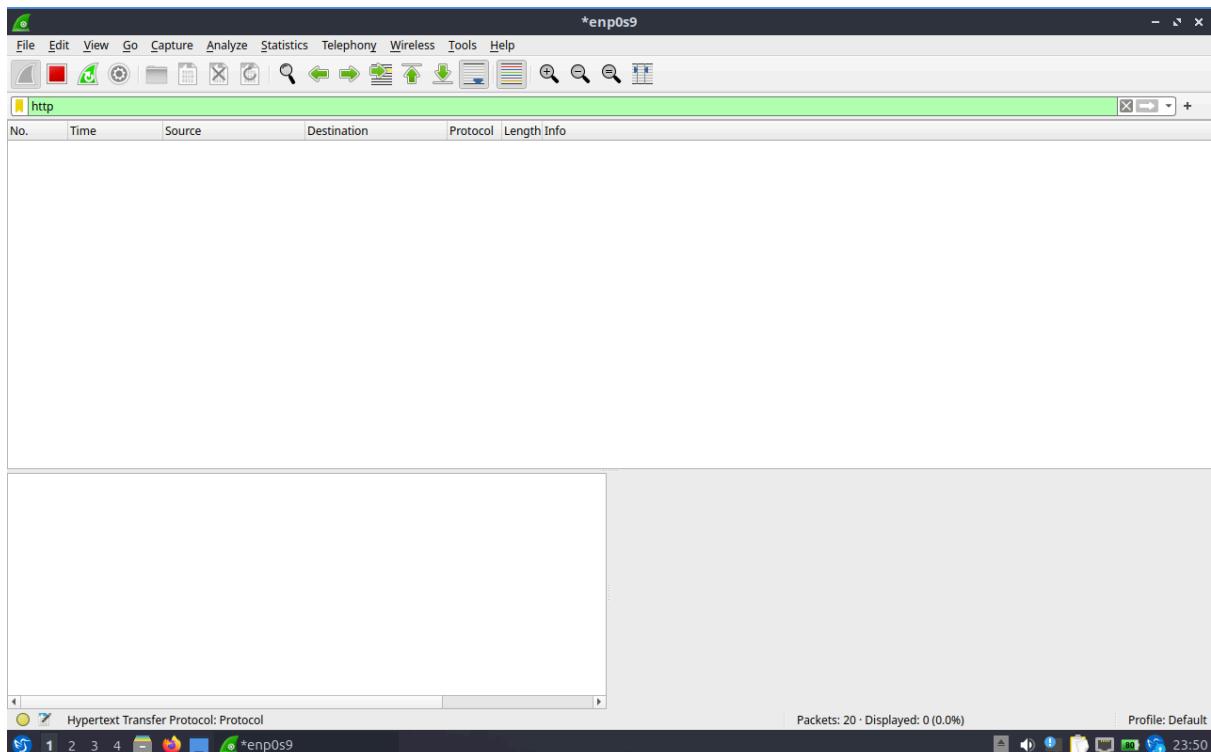
Sama seperti minggu lalu, kami akan menguji dengan memasukkan username: kelompok2 dan password:testuser. Lalu kami tekan enter, sembari mengecek di vm router untuk menganalisa paket traffic nya.



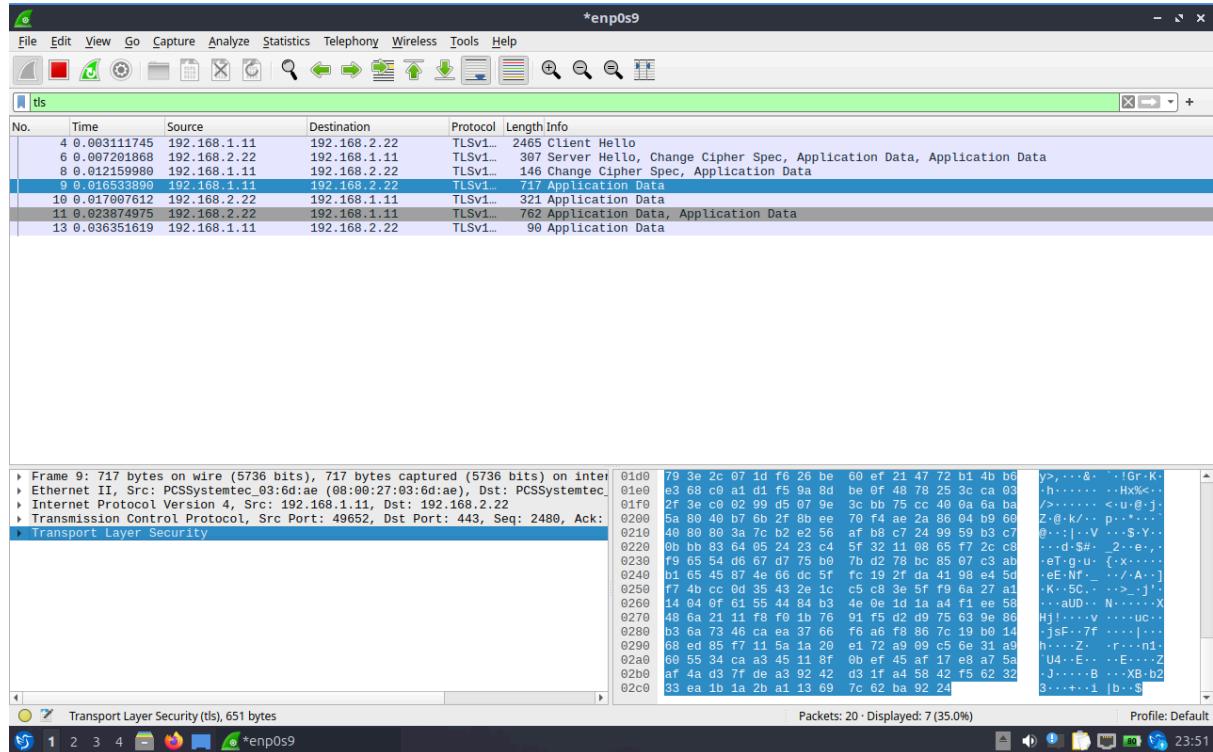
Sama seperti minggu lalu, kita akan menggunakan port enp0s9 karena port tersebut yang terhubung dengan vm server.



Nah, dapat terlihat kalau traffic sudah terpantau di port vm server, dan jika kita lihat pada filter protokol HTTP, maka hasilnya akan seperti ini.



Terbukti kosong, karena kita tidak memakai protokol itu lagi, namun HTTPS, dan ketika kita mengecek protokol tersebut, via filter tls, maka hasilnya akan seperti ini.



Nah, dapat kita lihat pada paket application data dari IP vm klien (192.168.1.11) yang bertujuan ke vm server (192.168.2.22), maka datanya akan beracak-acakan, karena data yang masuk itu sudah dienkripsi sehingga tidak lagi terlihat oleh pemantau pada traffic jaringan.

Pengujian minggu ini menunjukkan protokol HTTPS lebih aman dalam keperluan transfer data, terutama data sensitif seperti username dan password pengguna. Penggunaan protokol HTTPS akan mengenkripsi semua input yang dimasukkan pengguna, sehingga hanya penerima saja yang dapat membacanya, dan bukan pemantau jaringan.

Mungkin itu saja yang dapat kami sampaikan, kita ketemu lagi pada minggu depan saat kami membahas progress minggu ke lima: **Bandingkan perbedaan data yang terbaca**, yang kemungkinan akan dilakukan secara langsung di kelas. *See ya there, amigos, peace!*

- Kelompok 2