

# **LAPORAN AKHIR PROYEK**

**Judul Proyek:** Analisis Serangan Sniffing & Mitigasinya di Wireshark

**Kelompok:** 2 (Dua)

**Nama Mahasiswa / NIM:**

Dave Travis - 2201020008

Rizsky Parsadanta Rajagukguk - 2201020117

Muhammad Noval - 2201020014

Arya Winata - 2201020001

**Program Studi:** Teknik Informatika

**Mata Kuliah:** Perancangan Keamanan Sistem dan Jaringan

**Dosen Pengampu:** Ferdi Cahyadi, S.Kom., M.Cs.

**Semester / Tahun Ajaran:** 7 / 2025-2026

## PENDAHULUAN

### 1. Latar Belakang

Keamanan jaringan berperan sebagai fondasi utama dalam melindungi aspek kerahasiaan (*confidentiality*), keutuhan (*integrity*), dan ketersediaan (*availability*) informasi yang dikenal sebagai konsep CIA Triad [3]. Pada era digital, pertukaran data penting, seperti informasi autentikasi, data pribadi, hingga transaksi finansial, yang mana berlangsung secara intensif melalui jaringan lokal maupun internet [6]. Tanpa mekanisme pengamanan yang memadai, aliran data tersebut berpotensi dieksplorasi melalui serangan pasif, misalnya packet sniffing, yang memungkinkan pihak tidak berwenang memperoleh informasi dengan cara memantau lalu lintas jaringan.

Protokol HTTP, yang masih digunakan di banyak sistem warisan atau pengembangan internal, mengirimkan data dalam bentuk plain text, sehingga memudahkan penyerang untuk mencuri kredensial hanya dengan alat analisis seperti *Wireshark* [1] [6]. Di sisi lain, HTTPS yang mengenkripsi seluruh komunikasi melalui TLS/SSL menjadi standar keamanan modern. Namun, pemahaman mendalam tentang perbedaan teknis antara keduanya, serta efektivitas mitigasi seperti HSTS (*HTTP Strict Transport Security*), masih menjadi tantangan dalam pembelajaran keamanan jaringan [4]. Oleh karena itu, proyek ini dirancang untuk memberikan simulasi nyata yang menunjukkan risiko nyata dari penggunaan protokol tidak aman dan keandalan enkripsi sebagai lapisan pertahanan utama.

### 2. Tujuan Proyek

- a. Memahami cara kerja serangan *packet sniffing* dalam lingkungan jaringan tersegmentasi,
- b. Menganalisis kerentanan protokol HTTP terhadap ekstraksi data sensitif (seperti *username* dan *password*) melalui Wireshark,
- c. Membuktikan efektivitas mitigasi keamanan melalui penerapan HTTPS dan HSTS dalam melindungi data dari penyadapan pasif,
- d. Menyimulasikan skenario serangan realistik dengan penempatan *sniffer* sebagai pihak ketiga yang tidak terlibat dalam rute aktif, namun tetap mampu menangkap lalu lintas karena berada dalam subnet yang sama dengan target.

### 3. Ruang Lingkup

- a. Pembangunan topologi jaringan virtual menggunakan Oracle VirtualBox dengan empat mesin virtual: *Client*, *Router*, *Server*, dan *Sniffer*,
- b. Implementasi komunikasi multi-subnet dengan IP forwarding pada Router,
- c. Hosting halaman login sederhana di Server VM menggunakan Python HTTP/HTTPS server,
- d. Konfigurasi sertifikat SSL self-signed dan header HSTS pada layanan HTTPS,
- e. Penempatan Sniffer VM dalam subnet yang sama dengan Server untuk simulasi serangan *passive sniffing* yang realistik,

- f. Analisis lalu lintas HTTP dan HTTPS menggunakan Wireshark untuk membandingkan keterbacaan data sensitif.

Yang tidak termasuk dalam ruang lingkup proyek ini:

- a. Serangan aktif seperti Man-in-the-Middle (MITM) atau ARP spoofing,
- b. Penggunaan sertifikat SSL resmi dari otoritas sertifikat (CA),
- c. Implementasi HSTS dalam skenario berbasis domain (karena keterbatasan penggunaan alamat IP),
- d. Integrasi dengan sistem notifikasi eksternal (misalnya Telegram/WhatsApp),
- e. Pengujian di perangkat jaringan fisik (seperti MikroTik atau Cisco).

## DASAR TEORI

### 1. Keamanan Jaringan

Keamanan jaringan merupakan upaya sistematis untuk melindungi integritas, kerahasiaan, dan ketersediaan data yang dikirim melalui infrastruktur jaringan [3]. Dalam lingkungan yang terhubung, data sensitif seperti kredensial login rentan terhadap berbagai ancaman, termasuk penyadapan pasif (sniffing), modifikasi data, hingga pengalihan lalu lintas [6].

Prinsip dasar seperti CIA Triad (*Confidentiality, Integrity, Availability*) menjadi pondasi dalam merancang strategi pertahanan [3]. Salah satu pendekatan utama adalah penerapan enkripsi end-to-end, misalnya melalui protokol HTTPS yang memastikan bahwa meskipun lalu lintas berhasil diintersepsi, informasi di dalamnya tetap tidak dapat diakses oleh pihak tidak berwenang.

### 2. CIA Triad

Prinsip dasar keamanan informasi dikenal sebagai CIA Triad, yang terdiri dari tiga pilar utama [3]:

- a. *Confidentiality* (Kerahasiaan): Menjamin bahwa informasi hanya dapat diakses oleh pihak yang berwenang.
- b. *Integrity* (Integritas): Memastikan data tidak diubah secara tidak sah selama transmisi.
- c. *Availability* (Ketersediaan): Menjamin sistem dan data tersedia saat dibutuhkan.

Dalam konteks proyek ini, fokus utama terletak pada confidentiality, khususnya dalam melindungi kredensial pengguna (*username* dan *password*) selama transmisi melalui jaringan. Tanpa perlindungan yang memadai, data sensitif dapat diekstraksi oleh pihak ketiga melalui serangan *packet sniffing*, yang secara langsung melanggar prinsip kerahasiaan.

### 3. Packet Sniffing

*Packet sniffing* adalah teknik penyadapan jaringan yang digunakan untuk mengintersepsi, merekam, dan menganalisis lalu lintas data yang melewati suatu segmen jaringan [2]. Serangan ini dapat bersifat:

- a. Pasif: Penyerang hanya “mendengarkan” lalu lintas tanpa mengganggu komunikasi, yang mana akan sulit dideteksi.
- b. Aktif: Penyerang memodifikasi atau menyisipkan paket (misalnya melalui *ARP spoofing*), sifat ini terkenal lebih agresif namun berisiko terdeteksi.

Dalam proyek ini, simulasi dilakukan dalam mode pasif, di mana *Sniffer VM* ditempatkan dalam subnet yang sama dengan Server, memanfaatkan *promiscuous mode* untuk

menangkap seluruh lalu lintas yang melewati antarmuka jaringan lokal — merepresentasikan skenario nyata di jaringan bersama seperti kantor atau Wi-Fi publik.

#### 4. HTTP & HTTPS

HTTP (*HyperText Transfer Protocol*) adalah protokol aplikasi yang digunakan untuk mentransfer data web. Namun, HTTP mengirimkan seluruh data termasuk header, body, dan kredensial dalam bentuk teks biasa (*plain text*), sehingga rentan terhadap penyadapan [1].

Sebagai solusi, HTTPS (*HTTP Secure*) mengintegrasikan lapisan enkripsi TLS/SSL di atas HTTP [6]. Seluruh komunikasi enkripsi secara *end-to-end*, sehingga hanya klien dan server yang memiliki kunci privat dapat mendekripsi isi lalu lintas. Ini mencegah penyerang dari mengekstraksi data sensitif, bahkan jika mereka berhasil menangkap paket.

#### 5. Enkripsi TLS/SSL

Transport Layer Security (TLS), penerus dari SSL (*Secure Sockets Layer*), adalah protokol kriptografi yang menyediakan kerahasiaan melalui enkripsi simetris, autentikasi melalui sertifikat digital, dan integritas melalui *message authentication code* (MAC) [5].

Sertifikat *self-signed* digunakan untuk menyederhanakan implementasi, meskipun tidak diverifikasi oleh otoritas sertifikat (CA) resmi. Meski demikian, enkripsi tetap berfungsi penuh sehingga memastikan data tidak dapat dibaca oleh pihak ketiga.

#### 6. HTTP Strict Transport Security (HSTS)

*HTTP Strict Transport Security* (HSTS) merupakan mekanisme keamanan yang diimplementasikan melalui header HTTP untuk menginstruksikan browser agar secara eksklusif menggunakan protokol HTTPS ketika berkomunikasi dengan suatu domain [4]. Dengan mekanisme ini, browser secara otomatis menolak koneksi HTTP, termasuk ketika pengguna secara manual memasukkan awalan `http://`, sehingga risiko serangan downgrade dan man-in-the-middle dapat diminimalisirkan.

Namun, berdasarkan standar yang dijelaskan dalam RFC 6797, kebijakan HSTS tidak diterapkan pada koneksi berbasis alamat IP, misalnya 192.168.2.22, karena adanya potensi risiko keamanan seperti *spoofing*. Meskipun demikian, penerapan HSTS dalam penelitian ini tetap dilakukan sebagai bentuk mitigasi teoretis dan pembuktian konsep, dengan keberadaan serta konfigurasi header HSTS diverifikasi melalui fitur *Developer Tools* pada browser.

#### 7. Oracle VirtualBox

Oracle VirtualBox adalah platform virtualisasi yang memungkinkan pembuatan dan manajemen *virtual machine* (VM) pada satu sistem host fisik [7]. Dalam simulasi keamanan jaringan, VirtualBox memberikan lingkungan terisolasi namun fleksibel untuk

membangun topologi multi-perangkat, seperti *client*, *server*, *router*, bahkan penyerang (*sniffer*), tanpa memerlukan perangkat keras tambahan.

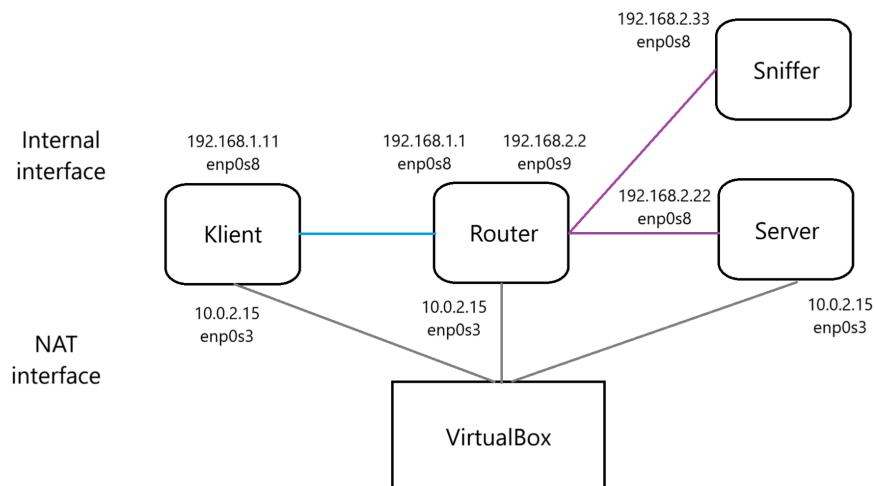
Fitur seperti Internal Network dan pengaturan Promiscuous Mode memungkinkan replikasi skenario nyata, termasuk serangan packet sniffing pasif, sehingga menjadi sarana ideal untuk eksperimen edukatif yang aman, terkendali, dan mudah direproduksi.

## 8. Wireshark

Wireshark adalah *network protocol analyzer open-source* yang memungkinkan pengguna untuk menangkap dan menganalisis lalu lintas jaringan secara *real-time* [7]. Dalam lingkungan virtual seperti Oracle VirtualBox, agar VM dapat menangkap lalu lintas yang bukan ditujukan kepadanya (seperti dalam serangan sniffing), promiscuous mode harus diaktifkan.

## PERANCANGAN SISTEM / ARSITEKTUR

Sistem yang dirancang dalam proyek ini terdiri dari empat mesin virtual (VM) yang dijalankan di atas platform *Oracle VirtualBox*, membentuk lingkungan simulasi jaringan multi-subnet yang merepresentasikan skenario serangan packet sniffing pasif. Topologi ini dirancang untuk memisahkan fungsi antara pengguna (*client*), penghubung (*router*), target (*server*), dan penyerang (*sniffer*), dengan tetap mempertahankan arsitektur yang realistik dan dapat direproduksi.



Setiap VM dikonfigurasi dengan dua *interface* jaringan:

- Internal Network*: untuk komunikasi antar-VM dalam lingkungan terisolasi.
- NAT Interface*: untuk akses internet dari host, namun karena proyek tugas ini memfokuskan pada *internal network* saja, maka penggunaan konfigurasi ini hanya terfokuskan pada instalasi awal saat persiapan semata.

### A. Client VM

- Peran: Pengguna yang melakukan permintaan login ke server.
- Interface 1 (Internal Network - net1):
  - Nama Interface: enp0s8
  - IP Address: 192.168.1.11/24
  - Default Gateway: 192.168.1.1 (Router)

3. Interface 2 (NAT):
  - a. Nama Interface: enp0s3
  - b. IP Address: 10.0.2.15 (DHCP dari VirtualBox)

#### B. Router VM

1. Peran: Menghubungkan dua subnet (192.168.1.0/24 dan 192.168.2.0/24) dan mengaktifkan IP forwarding.
2. Interface 1 (Internal Network - net1):
  - a. Nama Interface: enp0s8
  - b. IP Address: 192.168.1.1/24
3. Interface 2 (Internal Network - net2):
  - a. Nama Interface: enp0s9
  - b. IP Address: 192.168.2.1/24
4. Interface 3 (NAT):
  - a. Nama Interface: enp0s3
  - b. IP Address: 10.0.2.15

#### C. Server VM

1. Peran: Menyediakan layanan web HTTP/HTTPS tempat client melakukan login.
2. Interface 1 (Internal Network - net2):
  - a. Nama Interface: enp0s8
  - b. IP Address: 192.168.2.22/24
  - c. Default Gateway: 192.168.2.1 (Router)
3. Interface 2 (NAT):
  - a. Nama Interface: enp0s3
  - b. IP Address: 10.0.2.15

#### D. Sniffer VM

1. Peran: Penyerang pasif yang menangkap lalu lintas di subnet 192.168.2.0/24 tanpa menjadi bagian dari jalur komunikasi aktif.
2. Interface 1 (Internal Network - net2):
  - a. Nama Interface: enp0s8
  - b. IP Address: 192.168.2.33/24
  - c. Default Gateway: 192.168.2.1 (Router)
3. Interface 2 (NAT):
  - a. Nama Interface: enp0s3

- b. IP Address: 10.0.2.15

#### E. Alur Konfigurasi

1. Client - Server:

- Client mengirim permintaan ke <http://192.168.2.22> atau <https://192.168.2.22>.
- Karena alamat tujuan berbeda subnet, Client mengirim paket ke gateway-nya: 192.168.1.1 (Router).
- Router meneruskan paket ke subnet 192.168.2.0/24 via *interface* enp0s9.
- Paket mencapai Server (IP: 192.168.2.22) dan Sniffer juga menerima salinan paket karena berada di jaringan yang sama.

2. Server - Client:

- Server membalas ke gateway-nya: 192.168.2.1 (Router).
- Router meneruskan balasan ke Client via *interface* enp0s8.

VM	Interface	IP Address	Subnet	Peran
<b>Client</b>	enp0s8	192.168.1.11	192.168.1.0/24	Pengguna
	enp0s3	10.0.2.15	NAT	Akses Internet
<b>Router</b>	enp0s8	192.168.1.1	192.168.1.0/24	Gateway ke Client
	enp0s9	192.168.2.2	192.168.2.0/24	Gateway ke Server
	enp0s3	10.0.2.15	NAT	Akses Internet
<b>Server</b>	enp0s8	192.168.2.22	192.168.2.0/24	Web Server
	enp0s3	10.0.2.15	NAT	Akses Internet
<b>Sniffer</b>	enp0s8	192.168.2.33	192.168.2.0/24	Penyerang Pasif
	enp0s3	10.0.2.15	NAT	Akses Internet

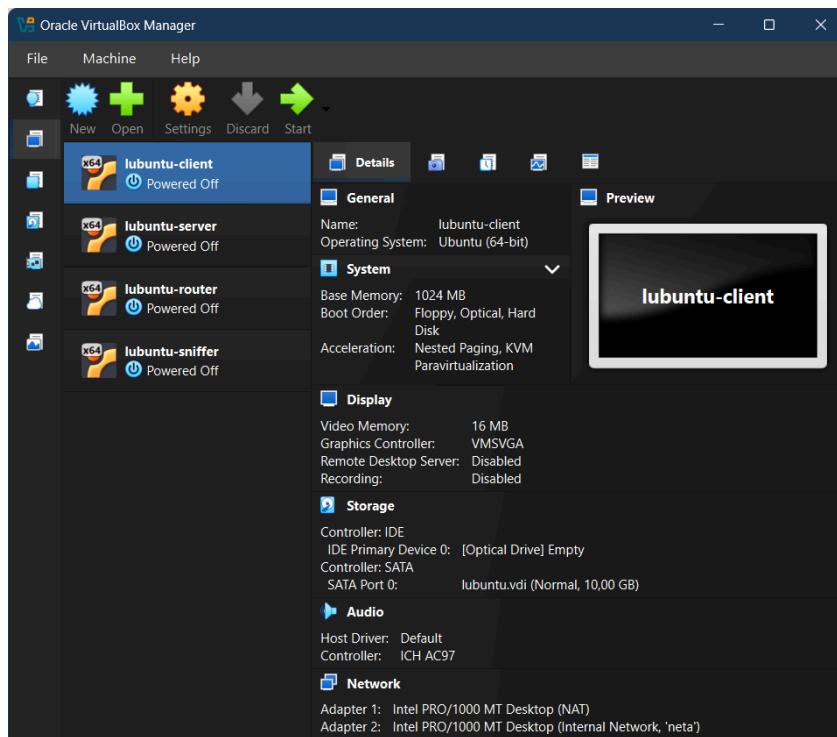
## IMPLEMENTASI

### 1. Konfigurasi Jaringan

4 unit *virtual machine* dibutuhkan untuk demonstrasi pembuatan jaringan LAN pada kali ini (*client, server, router, dan sniffer*), yang mana masing-masing vm terdapat IP masing-masing yang dapat dilihat pada gambar di atas. Pada jaringan LAN ini, dikarenakan terdapat pada VirtualBox, jadinya kami pisahkan menjadi 2 jaringan, yaitu jaringan internal, yang berkomunikasi dengan saluran enp0s8 dan enp0s9 yang IP nya akan diatur secara statis, dan jaringan NAT melalui enp0s3, yang mana menghubungkan tiap-tiap vm ke perangkat utama, alias laptop kelompok, via IP DHCP.

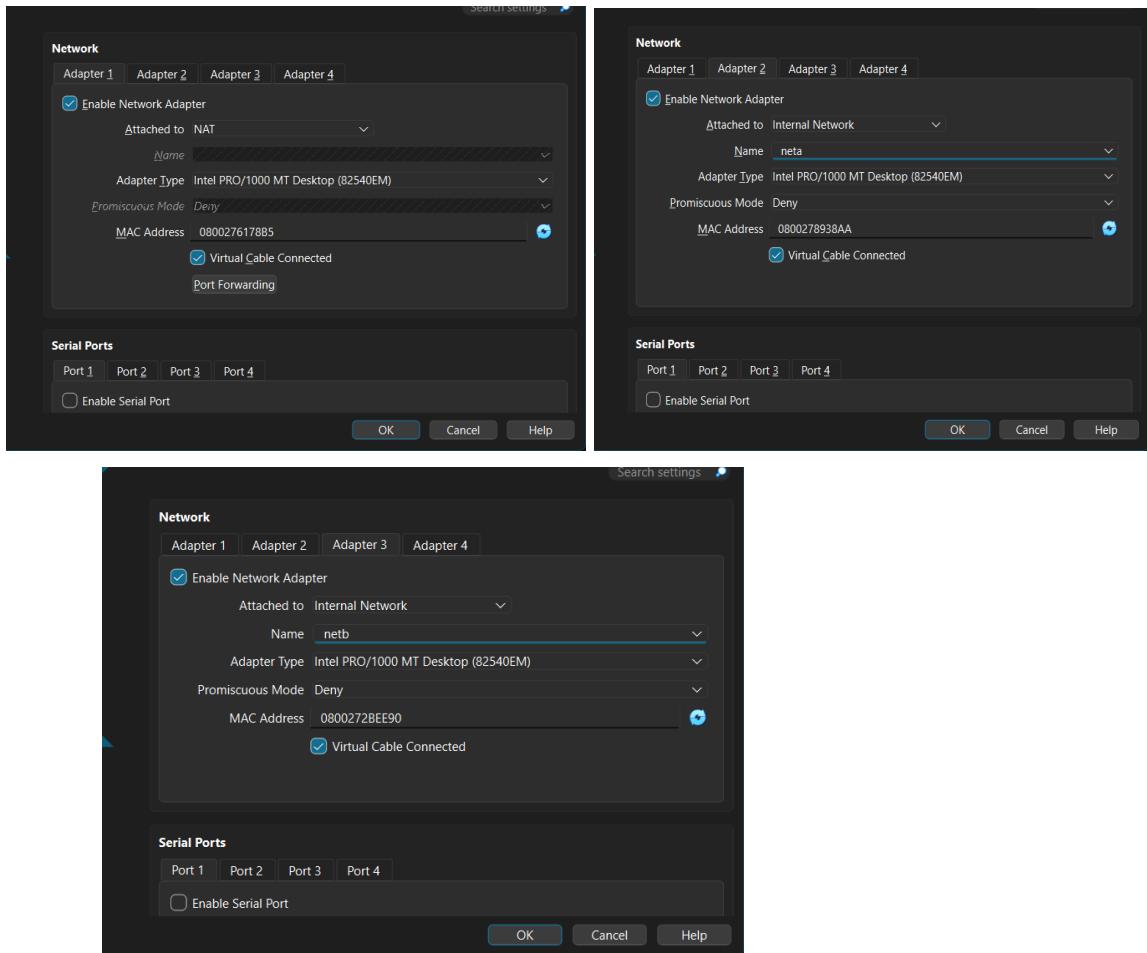
Untuk konfigurasi pada VirtualBox nya itu sendiri, dapat dilihat pada gambar di bawah beserta penjelasannya.

VirtualBox:



Karena adanya 4 perangkat yang kami ingin buat pada jaringan LAN ini, maka kami persiapkan 4 vm pada VirtualBox, menggunakan Linux dengan distro Lubuntu. Agar dapat membuat jaringan internal dan NAT seperti yang dibicarakan di atas, maka untuk masing-masing konfigurasi network vm ialah seperti ini.

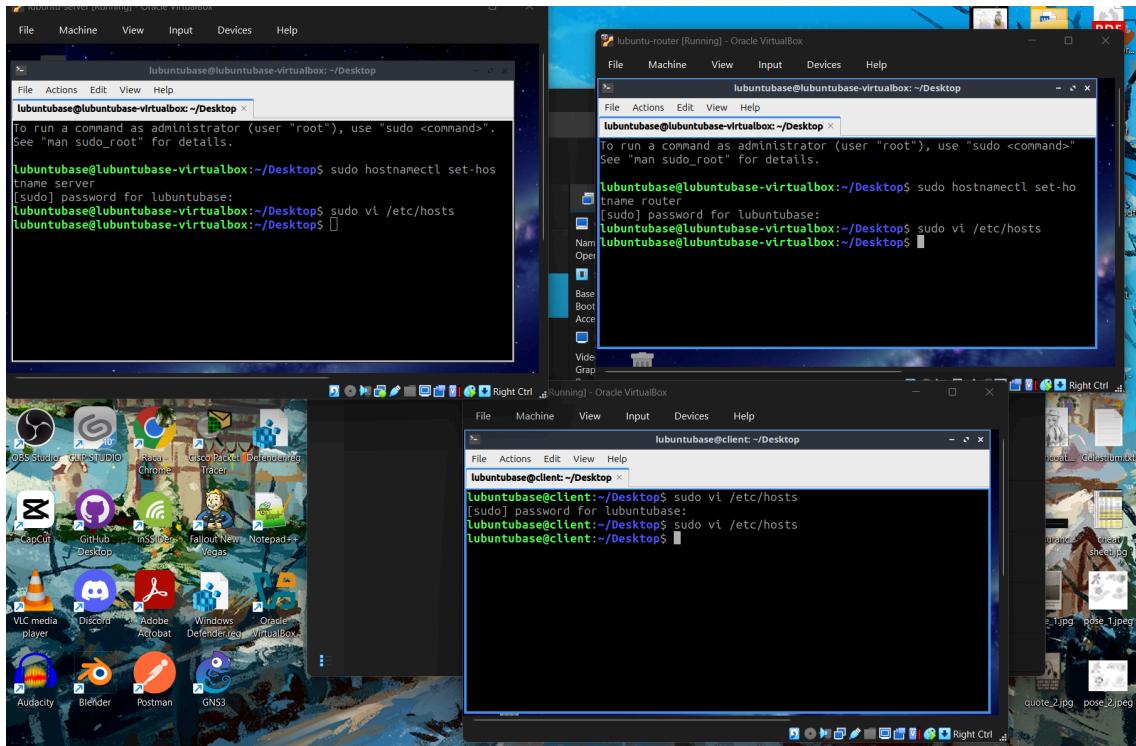
Kita mulai untuk vm client, yang mana akan menggunakan 2 slot adapter.



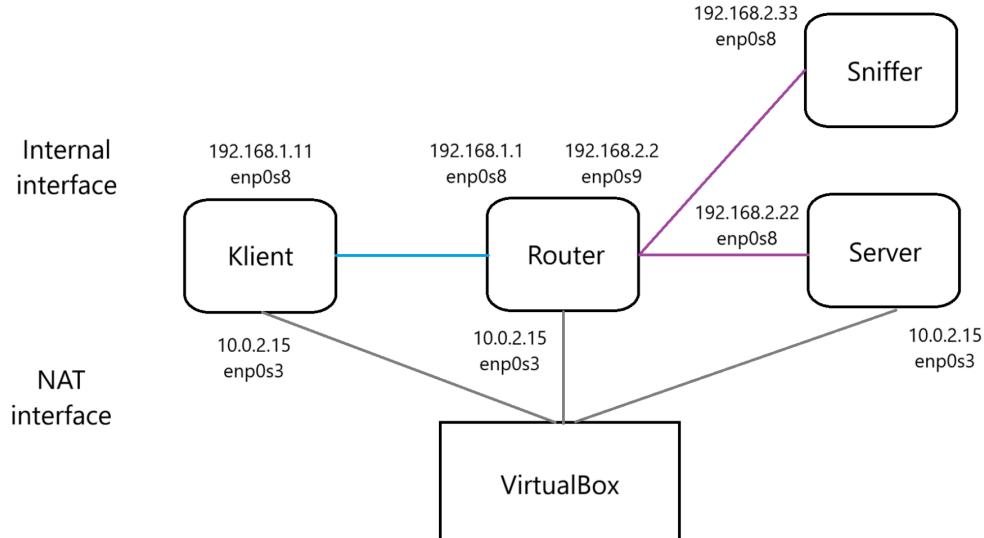
Pada adapter 1, kami membiarkan settingan default NAT, agar vm bisa tetap terhubung dengan perangkat utama, alias laptop, dan juga akan diatur sebagai default adapter bagi tiap-tiap vm nantinya, baik vm router dan vm server. Untuk adapter 2, kami mengaturnya sebagai Internal Network bernama ‘neta’, sehingga adanya jembatan antara vm klient dan vm router. Untuk Internal Network pada adapter 2 bernama ‘neta’ ini juga kami tambahkan pada vm router.

Untuk vm server, kami menambahkan adapter 3, yang mana juga Internal Network dengan nama ‘netb’ sebagai jembatan antara vm server dengan vm router. Hal yang sama juga kami tambahkan pada vm router. Untuk vm sniffer, konfigurasi yang sama juga diaplikasikan sebagaimana konfigurasi yang digunakan pada vm server, dikarenakan vm sniffer ialah replikasi dari pc sederhana.

Setelah selesai konfigurasi network pada halaman awal, kami boot up semua vm dan setup sederhana nama bagi masing-masing vm.



Setelah sudah di setup nama bagi masing-masing vm, maka akan kita bagi IP static untuk masing-masing vm, yang mana berdasarkan gambar di atas, kami tampilkan lagi.



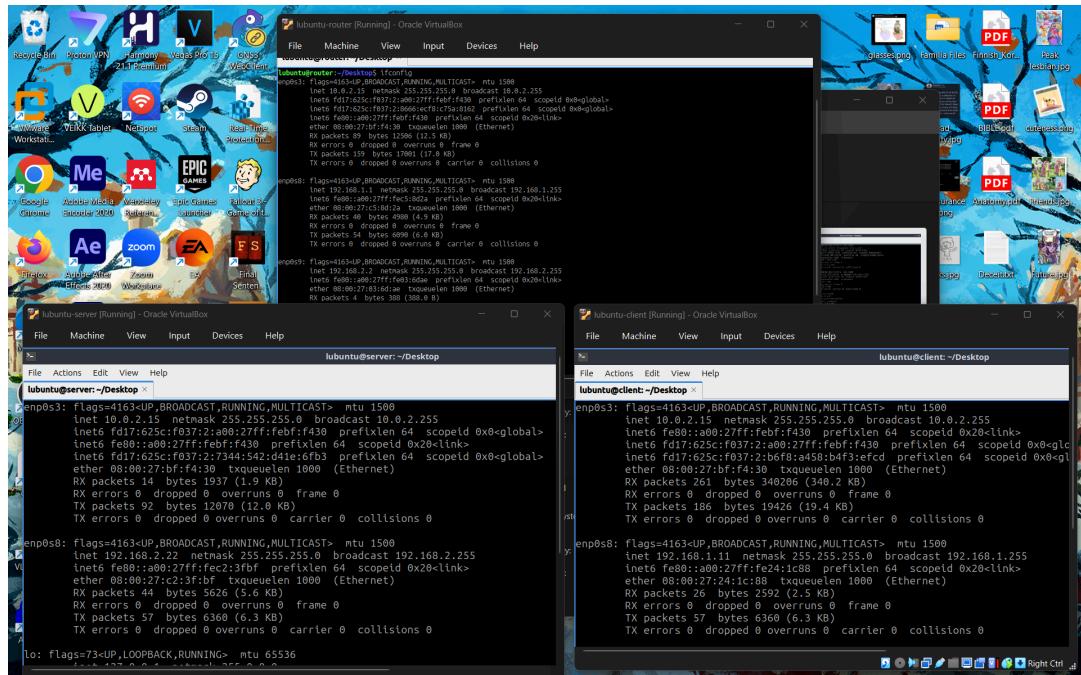
```
lubuntu@client: ~/Desktop
File Actions Edit View Help
lubuntu@client: ~/Desktop >
GNU nano 7.2                                     /etc/netplan/01-network-manager-all.yaml
# This file was written by calamares.
# Let NetworkManager manage all devices on this system.
# For more information, see netplan(5).
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp0s3:
      dhcp4: true

    enp0s8:
      dhcp4: false
      addresses:
        - 192.168.1.11/24
      routes:
        - to: 192.168.0.0/16
          via: 192.168.1.1

  nameservers:
    addresses: [8.8.8.8, 1.1.1.1]

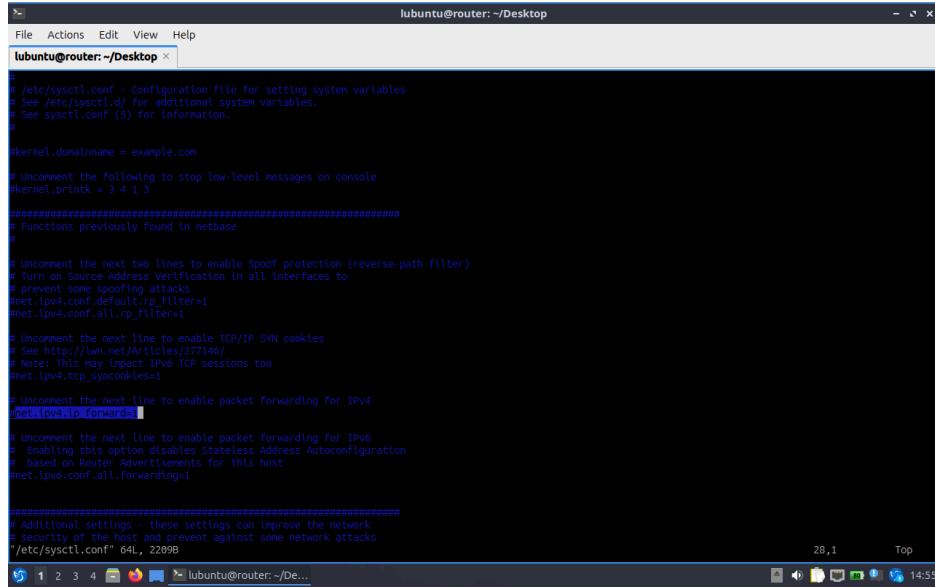
[ Read 20 lines ]  ^K Cut           ^T Execute   ^C Location   M-U Undo
^G Help          ^O Write Out     ^W Where Is   ^U Paste       ^J Justify   ^/ Go To Line M-E Redo
^X Exit          ^R Read File    ^V Replace
```

Gambar di atas ialah ilustrasi input IP statis pada vm client, yang mana kurang lebih sama juga penulisannya untuk vm lainnya. Dan jika sudah siap, maka hasilnya akan seperti ini.



IP address untuk masing-masing vm sudah diatur sesuai dengan rancangan.

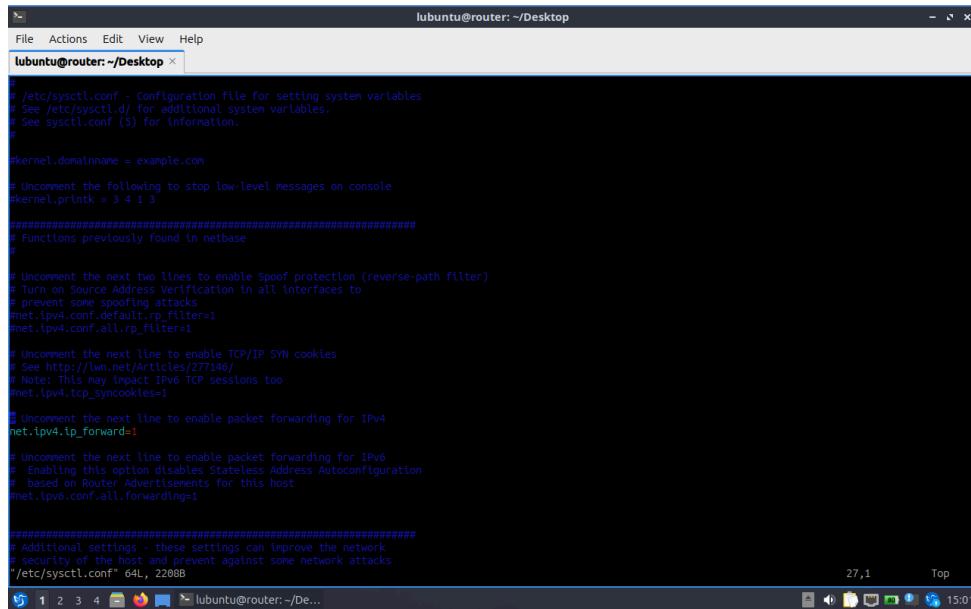
Sebelum kita testing ping untuk menguji apakah ketiga perangkat bisa berkomunikasi satu sama lain, kita perlu membuka file /etc/sysctl.yaml terlebih dahulu pada vm router, dikarenakan vm router ialah jembatan dari kedua perangkat, maka vm router perlu mengizinkan forward packet.



```
lubuntu@router: ~/Desktop
File Actions Edit View Help
lubuntu@router: ~/Desktop ×

# /etc/sysctl.conf - Configuration file for setting system variables
# See /etc/sysctl.d/ for additional system variables.
# See sysctl(5) for information.
#
#kernel.domainname = example.com
# Uncomment the following to stop low-level messages on console
#kernel.printk = 3 4 1 3
#####
# Functions previously found in netbase
#
# Uncomment the next two lines to enable Spoof protection (reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1
#
# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_synccookies=1
#
# Uncomment the next line to enable packet forwarding for IPv4
#net.ipv4.ip_forward=1
#
# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
# based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1
#####
# Additional settings: these settings can improve the network
# security of the host and prevent against some network attacks
"/etc/sysctl.conf" 64L, 2209B
28,1           Top
```

Tinggal dihapus simbol hastag nya, agar tidak dianggap sebagai komentar.



```
lubuntu@router: ~/Desktop
File Actions Edit View Help
lubuntu@router: ~/Desktop ×

# /etc/sysctl.conf - Configuration file for setting system variables
# See /etc/sysctl.d/ for additional system variables.
# See sysctl(5) for information.
#
#kernel.domainname = example.com
# Uncomment the following to stop low-level messages on console
#kernel.printk = 3 4 1 3
#####
# Functions previously found in netbase
#
# Uncomment the next two lines to enable Spoof protection (reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1
#
# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_synccookies=1
#
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
#
# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
# based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1
#####
# Additional settings: these settings can improve the network
# security of the host and prevent against some network attacks
"/etc/sysctl.conf" 64L, 2208B
27,1           Top
```

Oke, tinggal di simpan, dan kita coba ping dari vm client ke vm lainnya.

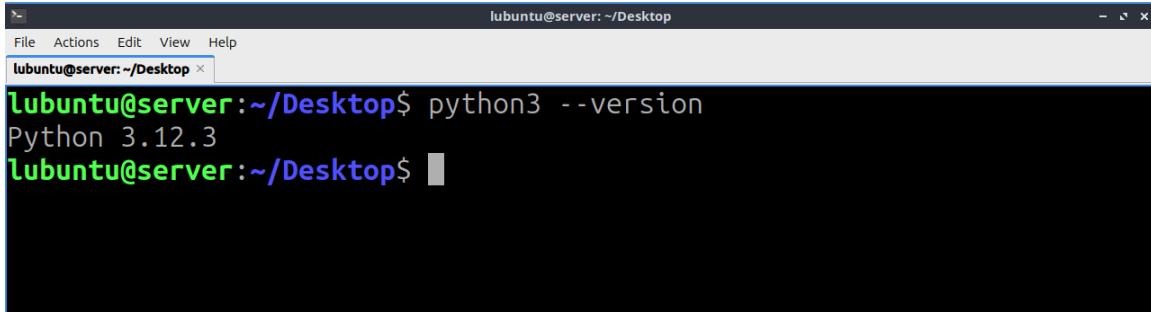
The screenshot shows a terminal window titled "lubuntu@client: ~/Desktop". The window contains the following command-line session:

```
lubuntu@client:~/Desktop$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=2.75 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.992 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=1.03 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=64 time=1.84 ms
^C
--- 192.168.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 0.992/1.652/2.745/0.716 ms
lubuntu@client:~/Desktop$ ping 192.168.2.2
PING 192.168.2.2 (192.168.2.2) 56(84) bytes of data.
64 bytes from 192.168.2.2: icmp_seq=1 ttl=64 time=0.924 ms
64 bytes from 192.168.2.2: icmp_seq=2 ttl=64 time=0.581 ms
64 bytes from 192.168.2.2: icmp_seq=3 ttl=64 time=1.25 ms
64 bytes from 192.168.2.2: icmp_seq=4 ttl=64 time=0.703 ms
^C
--- 192.168.2.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3056ms
rtt min/avg/max/mdev = 0.581/0.864/1.251/0.254 ms
lubuntu@client:~/Desktop$ ping 192.168.2.22
PING 192.168.2.22 (192.168.2.22) 56(84) bytes of data.
64 bytes from 192.168.2.22: icmp_seq=1 ttl=63 time=1.13 ms
64 bytes from 192.168.2.22: icmp_seq=2 ttl=63 time=2.14 ms
64 bytes from 192.168.2.22: icmp_seq=3 ttl=63 time=1.28 ms
64 bytes from 192.168.2.22: icmp_seq=4 ttl=63 time=1.06 ms
^C
--- 192.168.2.22 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 1.060/1.402/2.144/0.435 ms
lubuntu@client:~/Desktop$
```

Seperti yang terlihat pada gambar di atas, bahwa dari client, ping ke router dan ke server terbukti berhasil, dan menandakan bahwa jaringan LAN pada VirtualBox ini berhasil.

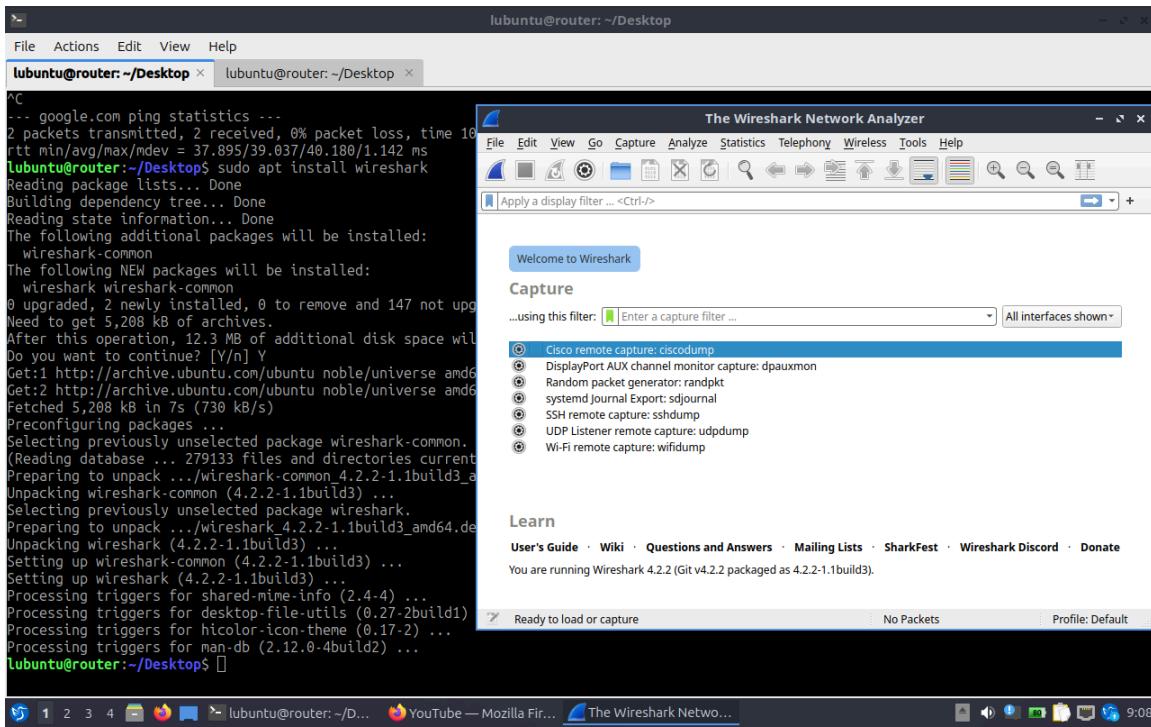
## 2. Instalasi Software

Sebelum dilakukannya pengujian, ada baiknya kami mempersiapkan terlebih dahulu tools-tools yang diperlukan pada tahapan ini pada seluruh vm.



```
lubuntu@server:~/Desktop$ python3 --version
Python 3.12.3
lubuntu@server:~/Desktop$
```

Pada vm server, kami memastikan bahwa Python sebagai media hosting website login HTTP sederhana sudah terinstall, walaupun Python sudah default pada OS Linux. Namun, tidak ada salahnya untuk *double check*, bukan?



```
lubuntu@router:~/Desktop$ sudo apt install wireshark
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  wireshark-common
The following NEW packages will be installed:
  wireshark wireshark-common
0 upgraded, 2 newly installed, 0 to remove and 147 not upgraded.
Need to get 5,208 kB of archives.
After this operation, 12.3 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://archive.ubuntu.com/ubuntu noble/universe amd64 wireshark-common 4.2.2-1.1build3_amd64.deb [5,208 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble/universe amd64 wireshark 4.2.2-1.1build3_amd64.deb [12.3 kB]
Fetched 5,208 kB in 7s (730 kB/s)
Preconfiguring packages ...
Selecting previously unselected package wireshark-common.
(Reading database ... 279133 files and directories currently installed)
Preparing to unpack .../wireshark-common_4.2.2-1.1build3_amd64.deb ...
Unpacking wireshark-common (4.2.2-1.1build3) ...
Selecting previously unselected package wireshark.
Preparing to unpack .../wireshark_4.2.2-1.1build3_amd64.deb ...
Unpacking wireshark (4.2.2-1.1build3) ...
Setting up wireshark-common (4.2.2-1.1build3) ...
Setting up wireshark (4.2.2-1.1build3) ...
Processing triggers for shared-mime-info (2.4-4) ...
Processing triggers for desktop-file-utils (0.27-2build1) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for man-db (2.12.0-4build2) ...
lubuntu@router:~/Desktop$
```

The Wireshark Network Analyzer application is running in the background, showing its main interface with various capture options and a packet list.

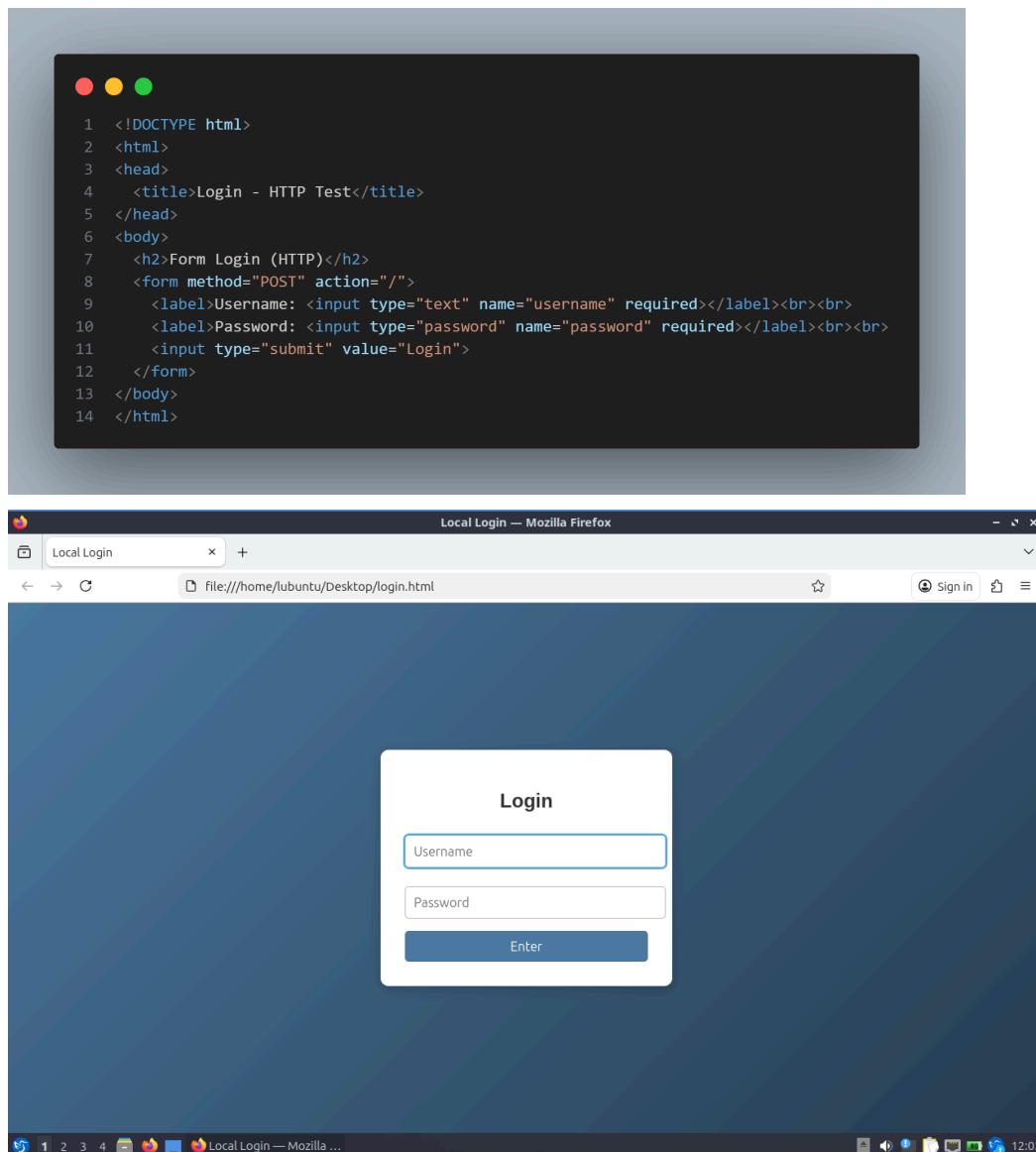
Lalu, pada vm sniffer, kami menginstall Wireshark yang akan digunakan untuk memantau traffic pada percobaan kali ini.

Dan pada vm client, hanya diperlukan *browser* saja untuk percobaan kali ini, dan kami akan menggunakan Firefox sebagai *browser* bawaan dari Lubuntu.

## PENGUJIAN DAN ANALISIS

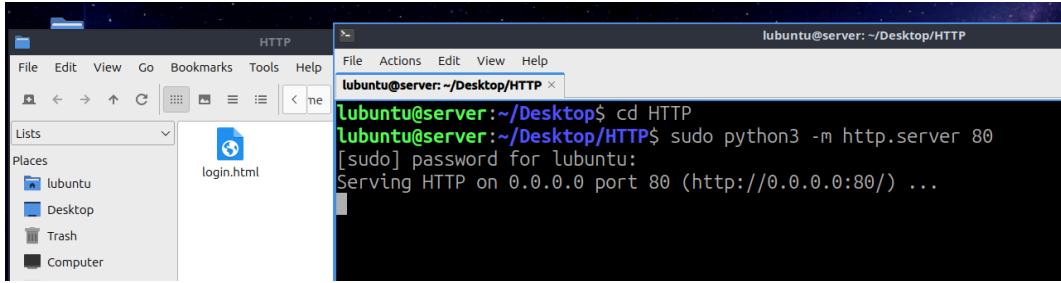
### A. Capture dan Analisis Paket Login HTTP

Sebelum melakukan capture dan analisis paket login HTTP, diperlukannya media pengujian, maka pada vm server diperlukannya *website* sederhana yang menyediakan website lokal yang mendukung sistem POST login, sehingga bisa dianalisis *traffic* nya, maka kami membuat nya seperti ini:



Nah, karena halaman login.html sudah siap, maka saat nya kami menghosting secara lokal dari vm server, sehingga nantinya kalau sudah up, bisa diakses oleh vm klien, karena sudah terhubung via *interface* lokal. Dikarenakan

pada vm server tadi sudah terinstall python, maka tinggal dijalankan web server nya berdasarkan direktori file login.html nya, dan filenya pun sudah siap untuk diakses dari vm klien.

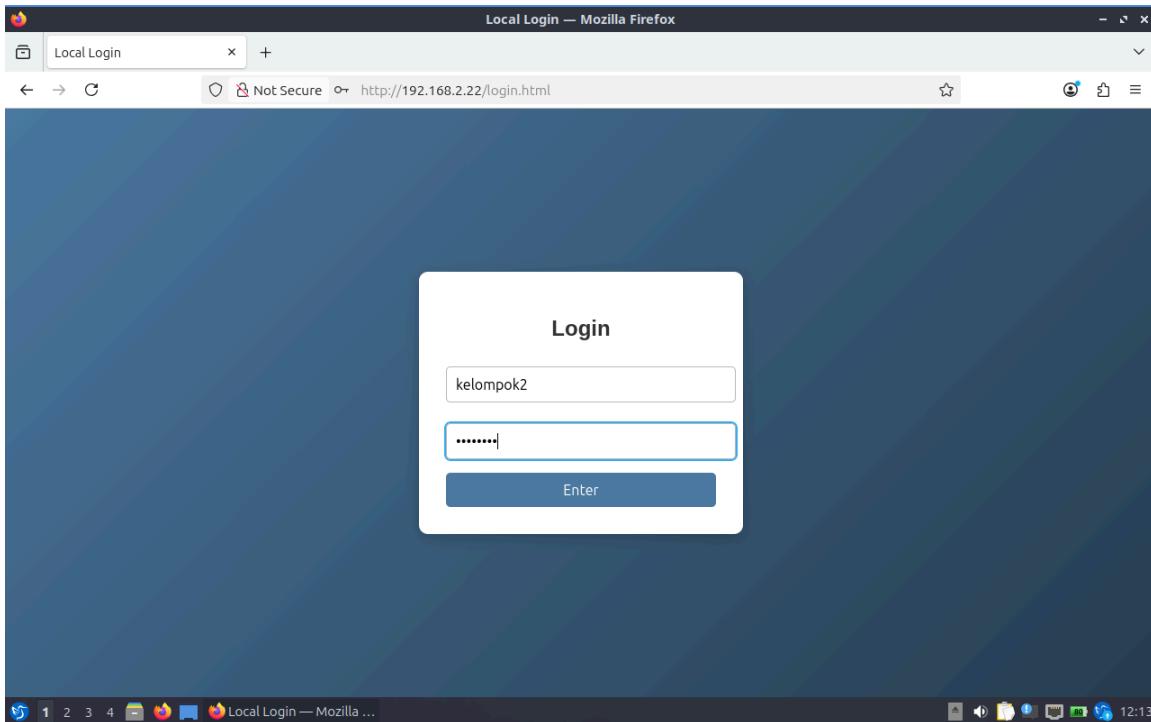


Setelah server siap menjalankan web server dari file login.html tadi, maka kita berpindah ke vm klien dan membuka laman IP address dari vm server, yaitu 192.168.2.22.

The top screenshot shows a Mozilla Firefox window titled "Directory listing for / — Mozilla Firefox". The address bar says "Not Secure http://192.168.2.22". Below the address bar, it says "Directory listing for /". Underneath, there is a link to "login.html".

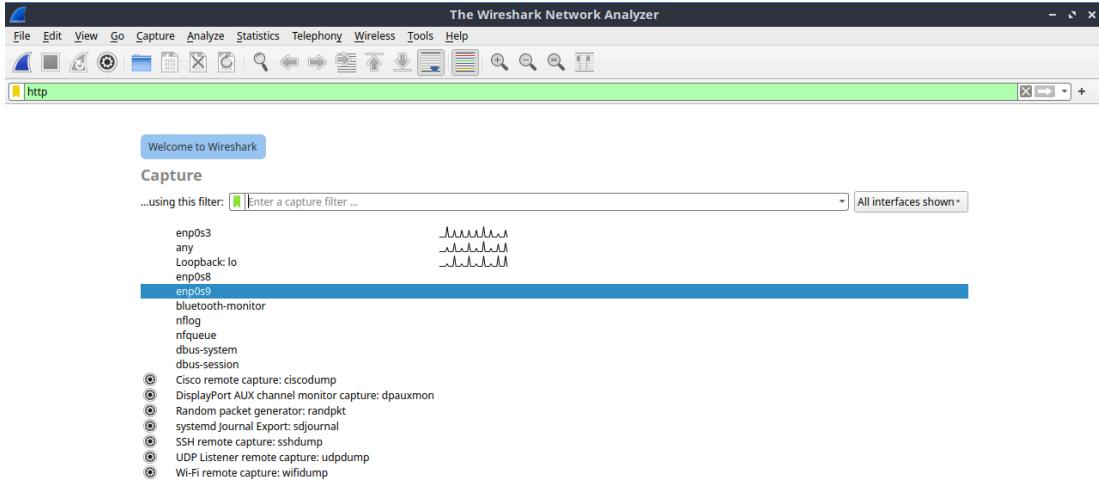
The bottom screenshot shows a Mozilla Firefox window titled "Local Login — Mozilla Firefox". The address bar says "Not Secure http://192.168.2.22/login.html". Below the address bar, it says "Local Login". A "Login" form is centered on the screen, containing fields for "Username" and "Password" and a blue "Enter" button.

Nah, sudah terlihat halaman loginnya dari vm klien, maka tinggal kita coba login saja.



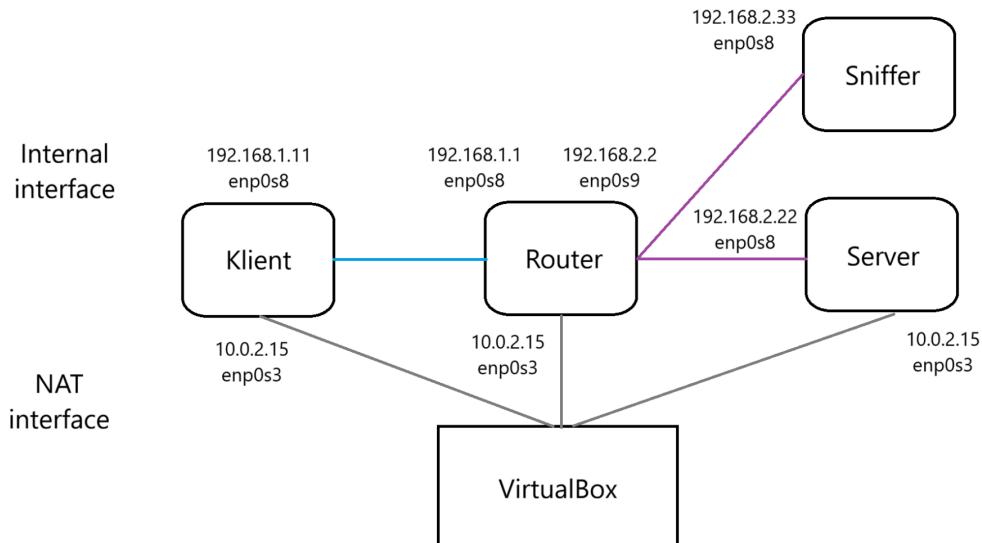
Kami mencoba untuk memasukkan username: kelompok2, password: testuser. Dan setelah siap diisi, maka kami menekan tombol enter, sehingga data dari vm klien dapat masuk ke dalam jaringan vm server, dengan dibuktikan dari pengecekan via Wireshark di vm sniffer, yang mana akan kami jelaskan di bawah.

VM sniffer secara eksklusif kami buatkan secara mandiri sebagai ilustrasi pihak penyusup, yang mana bisa saja berada dalam satu jaringan dengan perangkat yang sedang melakukan transfer data, dikarenakan pihak sniffer ini pastinya memiliki jejak transfer data pada perangkat-perangkat yang berada pada satu jaringan dengan ia.

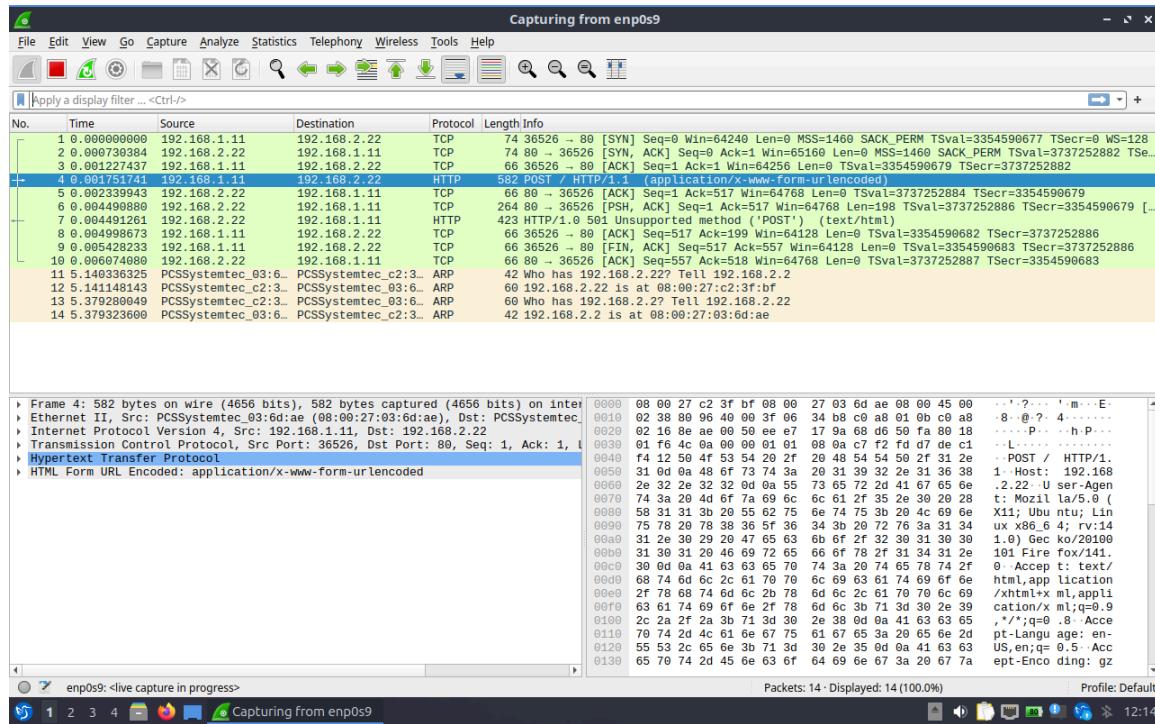


Dan bisa dilihat ada beberapa port yang dapat kita *capture traffic* nya, namun berdasarkan dari topologi jaringan kami:

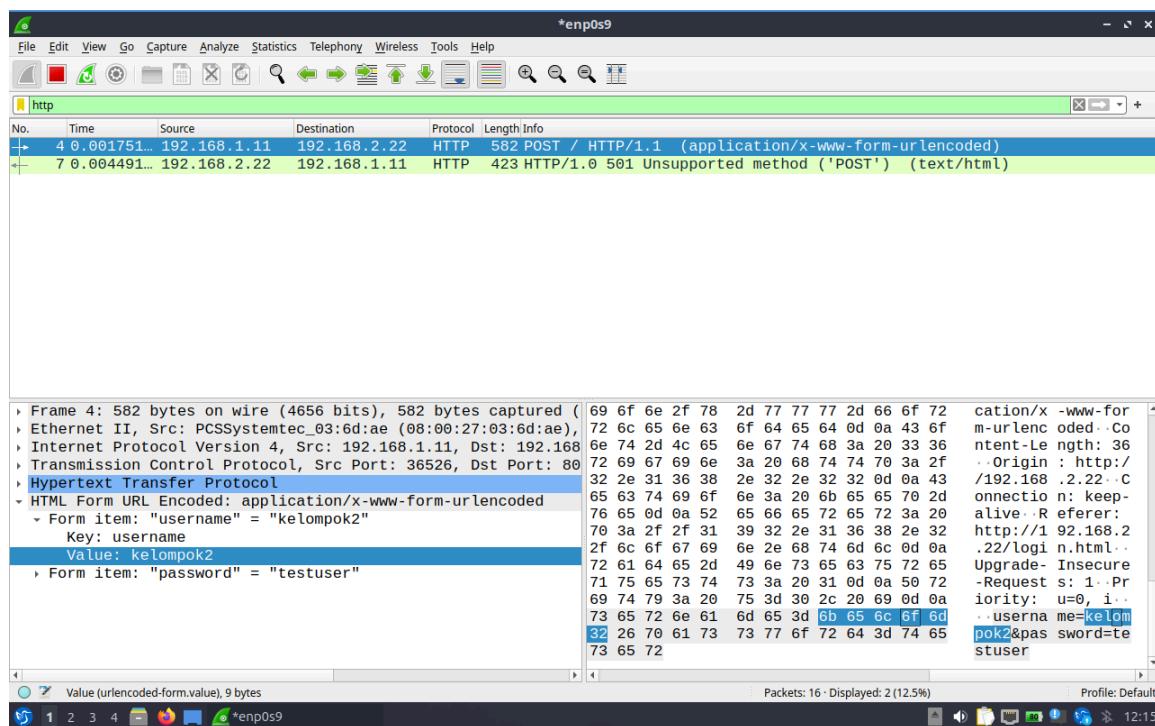
Desain Topologi:



Dapat dilihat kalau yang menghubungkan antara port vm router dan vm server (sebagai pemilik web server) ialah enp0s9, sehingga port ini lah yang akan kita pilih untuk di *capture traffic* nya.



Sudah terlihat kalau ada beberapa paket yang masuk ke dalam vm server, namun karena target kita minggu ini ialah HTTP, maka kita akan filter berdasarkan HTTP.

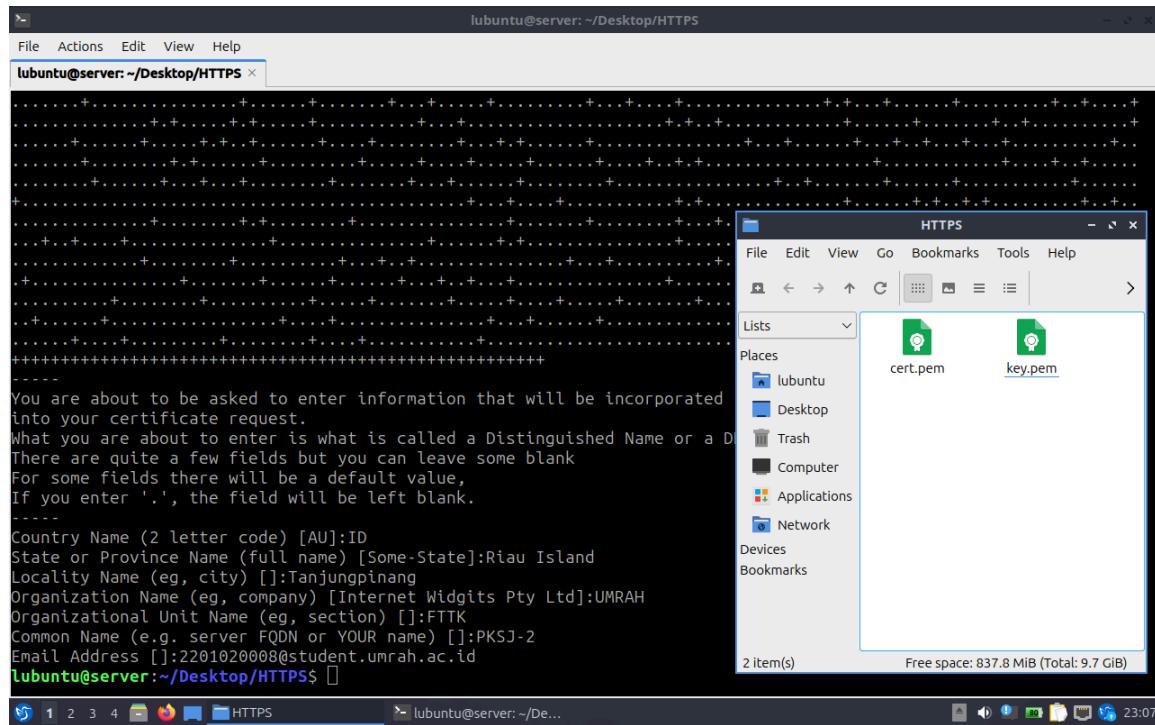


Nah, sudah terlihat bukan, bahwa protokol HTTP yang masuk ke dalam vm server via POST dari IP source 192.168.1.11, yaitu vm client (berdasarkan topologi di atas), terlihat plaintext nya yang mengandung text username dan password yang dimasukkan.

Hal ini membuktikan bahwa protokol HTTP yang sering digunakan terlihat tidak aman, karena perangkat yang tidak ada keterlibatan dalam transfer data, dan hanya berada di jaringan yang sama, alias *man in the middle*, dapat dengan mudahnya mendeteksi dan melihat isi dari data transferan antara dua belah pihak.

## B. Capture dan Analisis Paket Login HTTPS

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()
```

```
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()

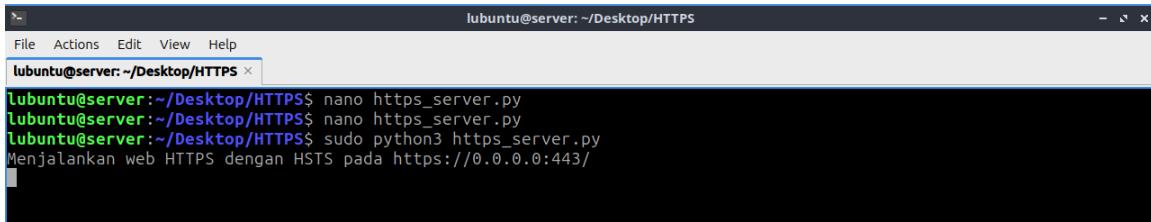
context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
context.load_cert_chain(certfile="cert.pem", keyfile="key.pem")

# 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()

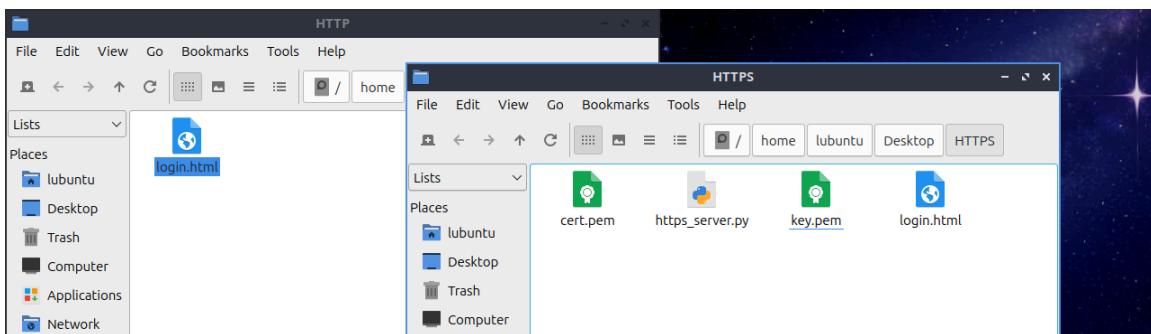
^G Help          ^O Write Out      ^W Where Is      ^K Cut           ^T Execute       ^C Location      M-U Undo
^X Exit          ^R Read File      ^Y Replace       ^U Paste         ^J Justify       ^I Go To Line    M-E Redo
```

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



```
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.



## Directory listing for /

- [cert.pem](#)
- [https\\_server.py](#)
- [key.pem](#)
- [login.html](#)

Local Login — Mozilla Firefox

Local Login | 192.168.2.22/login.html

Login

Username

Password

Enter

Network Tab (1 request)

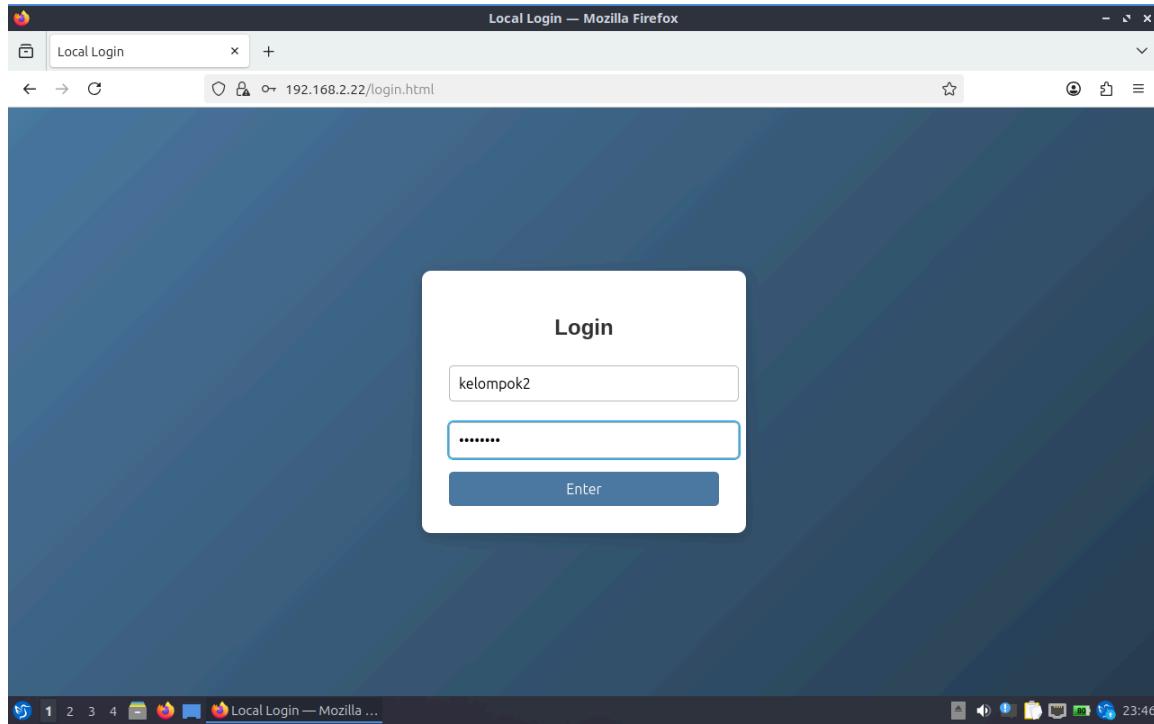
Status	Meth...	Domain	File	Initiator	Type	Transferred	Size
200	GET	192.168.2.22	login.html	document	HTML	cached	1.75 kB

Response Headers (201 B)

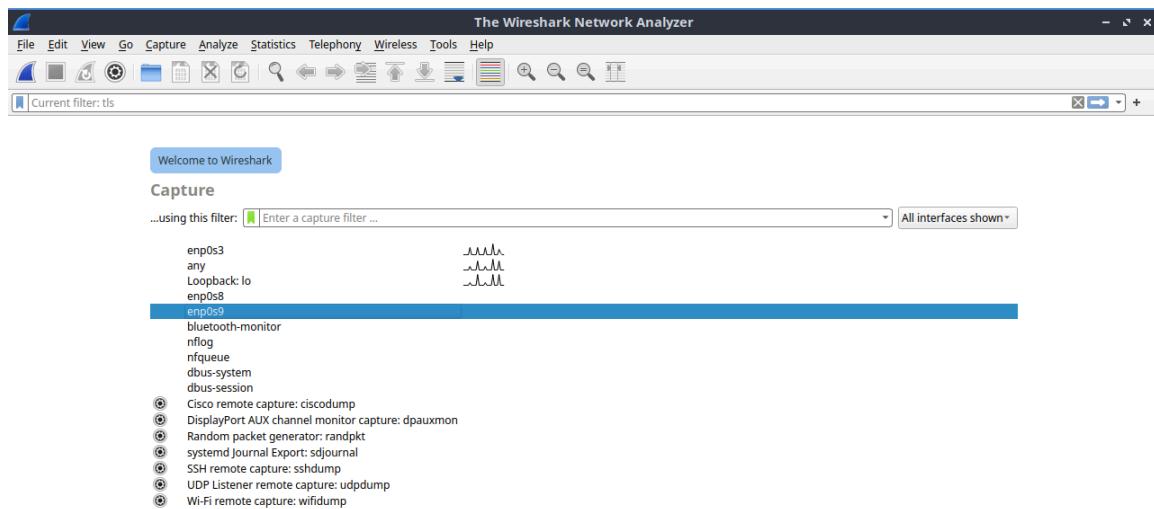
- Date: Wed, 03 Dec 2025 16:44:43 GMT
- Server: SimpleHTTP/0.6 Python/3.12.3
- Strict-Transport-Security: max-age=31536000; includeSubDomains
- X-Content-Type-Options: nosniff

2 requests | 2.09 kB / 0 B transferred | Finish: 538 ms | DOMContentLoaded: 351 ms | load: 371 ms

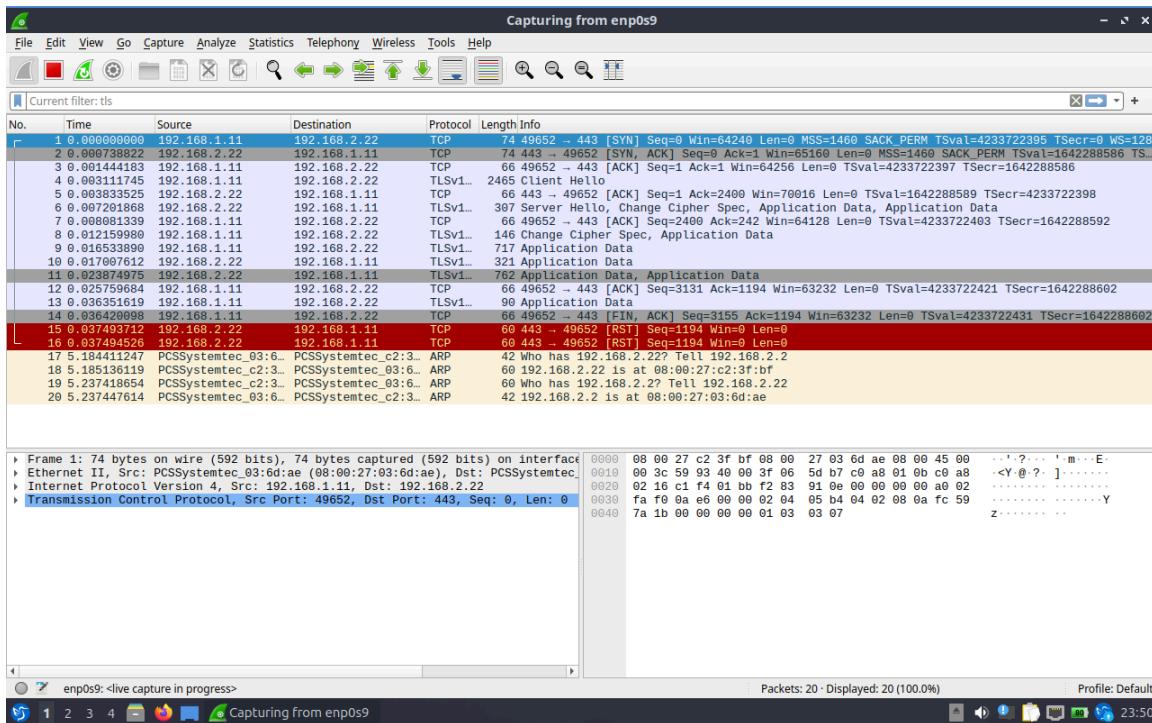
Oke, sudah terlihat halaman login bisa diakses, dan jika kita lihat juga pada *inspect network header*, pada *response headers* nya terlihat kalau konfigurasi HSTS kita sudah berjalan. Dan kita lanjutkan proses login kita.



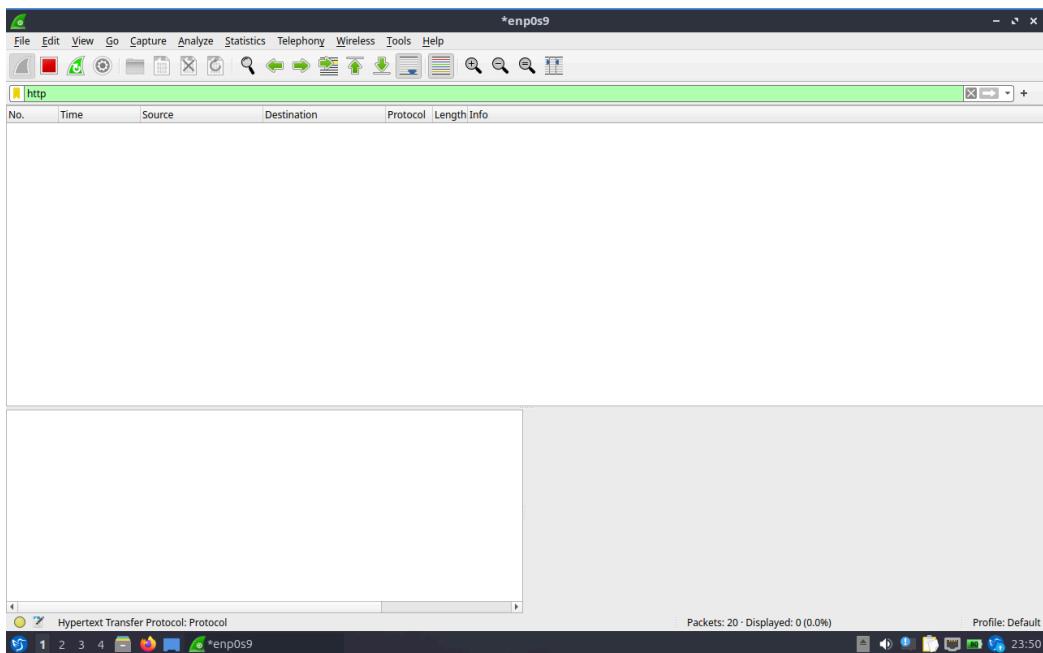
Sama seperti minggu lalu, kami akan menguji dengan memasukkan username: kelompok2 dan password:testuser. Lalu kami tekan enter, sembari mengecek di vm router untuk menganalisa paket traffic nya.



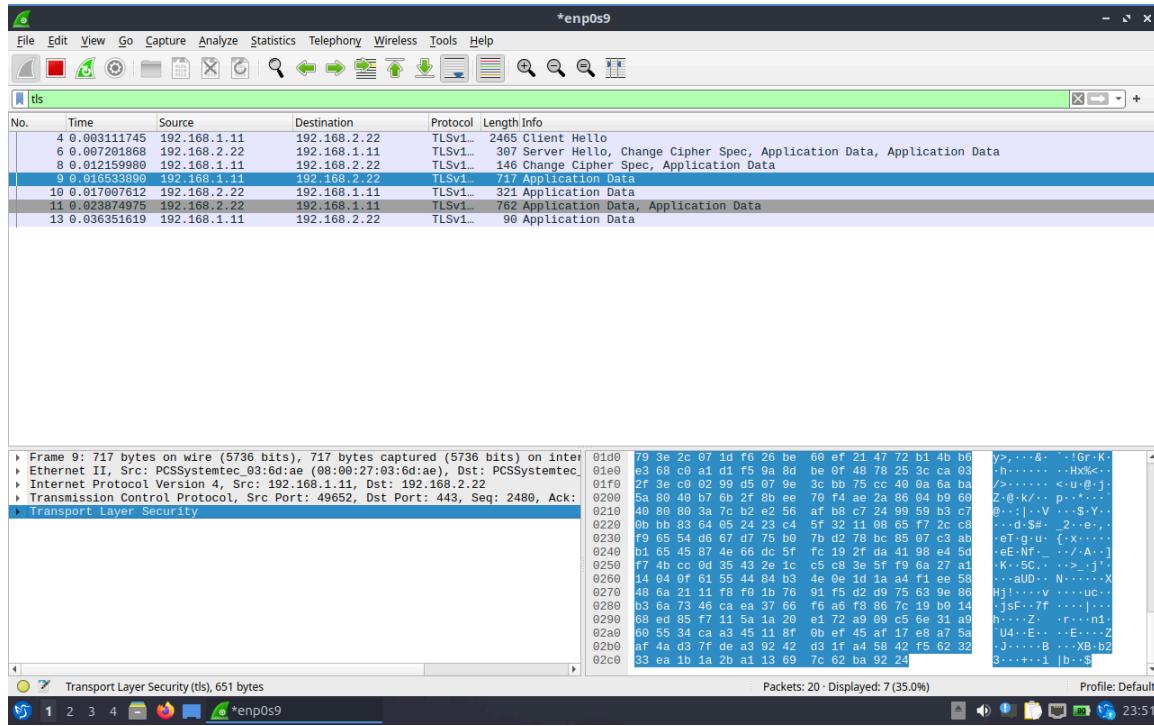
Sama seperti minggu lalu, kita akan menggunakan port enp0s9 karena port tersebut yang terhubung dengan vm server.



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 client (192.168.1.11) yang bertujuan ke vm server (192.168.2.22), maka datanya akan ber acak-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.

## KESIMPULAN

### 1. Pencapaian proyek

Proyek berhasil membangun simulasi serangan *packet sniffing* pasif dalam lingkungan virtual menggunakan VirtualBox, serta membuktikan secara eksplisit bahwa protokol HTTP mengirimkan data sensitif (seperti *username* dan *password*) dalam bentuk *plain text*, sedangkan HTTPS melindungi data tersebut melalui enkripsi TLS yang tidak dapat dibaca oleh pihak ketiga.

### 2. Pembelajaran yang diperoleh

Kami memperoleh pemahaman mendalam mengenai prinsip keamanan jaringan berbasis *confidentiality*, cara kerja enkripsi end-to-end, mekanisme HSTS, serta teknik analisis lalu lintas jaringan menggunakan Wireshark dalam skenario serangan realistik.

### 3. Kelemahan atau batasan sistem

Simulasi menggunakan alamat IP (bukan domain), sehingga fitur HSTS tidak dapat diuji secara fungsional; selain itu, sertifikat SSL yang digunakan bersifat *self-signed*, dan lingkungan virtual tidak merepresentasikan kompleksitas jaringan fisik seperti *latency*, *noise*, atau keberadaan *switch/bridge* nyata.

## SARAN PENGEMBANGAN

### 1. Implementasi dengan Domain Lokal

Agar fitur HSTS dapat diuji secara fungsional, sistem dapat dikembangkan dengan menggunakan domain lokal (misalnya server.local) melalui konfigurasi file hosts atau DNS internal. Hal ini memungkinkan *browser* menerapkan kebijakan HSTS secara penuh, yang tidak berlaku saat menggunakan alamat IP langsung.

### 2. Integrasi Notifikasi Otomatis:

Sistem dapat diperluas dengan menambahkan skrip pemantauan lalu lintas HTTP yang mengirimkan notifikasi *real-time* (misalnya ke Telegram atau WhatsApp) ketika terdeteksi transmisi data sensitif tanpa enkripsi sebagai bentuk *early warning system* untuk administrator jaringan.

### 3. Pengujian di Jaringan Fisik:

Simulasi dapat direplikasi di lingkungan jaringan nyata menggunakan perangkat seperti MikroTik atau Raspberry Pi, untuk menguji ketahanan terhadap faktor dunia nyata seperti *broadcast domain*, *switch behavior*, dan *network noise*.

### 4. Penambahan Serangan Aktif:

Proyek ini kedepannya dapat dikembangkan dengan mensimulasikan serangan *Man-in-the-Middle* (MITM) aktif menggunakan alat seperti *ettercap* atau *arpspoof*, untuk membandingkan efektivitas mitigasi terhadap serangan pasif vs aktif.

### 5. Penggunaan Sertifikat SSL Resmi:

Untuk simulasi yang lebih mendekati produksi, sistem dapat dikonfigurasi dengan sertifikat dari *Let's Encrypt* atau *CA internal*, sehingga peringatan keamanan di browser dapat dihilangkan dan pengalaman pengguna lebih realistik.

## DAFTAR PUSTAKA

- [1] R. T. Fielding, M. Nottingham, and J. Reschke, “RFC 9110: HTTP Semantics,” 2022. [Online]. Available: <https://www.rfc-editor.org/info/rfc9110>
- [2] G. Joseph, J. Osamor, and F. Olajide, “A Systematic Review of Network Packet Sniffing Tools for Enhancing Cybersecurity in Business Applications,” *International Journal of Intelligent Computing Research*, vol. 15, no. 1, pp. 1292–1307, Jun. 2024, doi: 10.20533/ijicr.2042.4655.2024.0157.
- [3] B. Setya Putra and D. B. Santoso, “Analisis Keamanan Website Berbasis WordPress melalui Penetration Testing untuk Meningkatkan Keamanan Digital,” 2025.
- [4] M. Nieles, K. Dempsey, and V. Y. Pillitteri, “An introduction to information security,” Gaithersburg, MD, Jun. 2017. doi: 10.6028/NIST.SP.800-12r1.
- [5] N. Dharmawan, Gani Indriyanta, and I Kadek Dendy Senapartha, “ANALISIS KEAMANAN JARINGAN UNIVERSITAS KRISTEN DUTA WACANA DENGAN SERANGAN SSL/TLS,” *Jurnal Terapan Teknologi Informasi*, vol. 6, no. 2, pp. 121–130, Oct. 2022, doi: 10.21460/jutei.2022.62.214.
- [6] J. Zhang, H. Feng, B. Liu, and D. Zhao, “Survey of Technology in Network Security Situation Awareness,” Mar. 01, 2023, *MDPI*. doi: 10.3390/s23052608.
- [7] “White Paper / Oracle VM VirtualBox Overview / Version 2.0 Oracle VM VirtualBox Overview,” 2021.