BAB 5 IMPLEMENTASI

5.1 Implementasi

Implementasi dilakukan dengan cara mengikuti desain yang telah dibuat. Untuk menerapkan sistem tersebut diperlukan suatu perangkat keras dan instalasi perangkat lunak serta konfigurasi komponen-komponen yang dibutuhkan dalam membangun sistem.

5.1.1 Spesifikasi Sistem

Untuk melakukan implementasi dibutuhkan beberapa komponen pada sebuah PC untuk menunjang keberhasilan sistem. Berikut ini merupakan komponen yang diperlukan untuk implementasi:

Prosesor	: Processor Intel Core i3-2370M CPU 2.40GHz
Hardisk	: 500 GB
Memory	: 8 GB
Sistem Operasi	: Ubuntu 14.04

5.2 Implementasi Sistem

Imlementasi dari laboratorium virtual jaringan komputer dibutuhkan komponen pendukung untuk keperluan fitur dalam sistem.

5.2.1 *Docker*

Pada penelitian ini *docker* di install dalam personal komputer. *Docker* yang digunakan pada penelitian ini versi 1.9.1. untuk melakukan instalasi *packet docker* dapat menggunakan perintah *apt-get install docker-engine* pada *terimanl* ubuntu kemudian lakukan *update* pada *repository* di dalam *image* ubuntu supaya komponen-kompinen *docker terupdate* dengan baik.

Langkah selanjutnya adalah pembuatan serta konfigurasi sesuai materi yang dibutuhkan untuk melakukan praktikum jaringan komputer. Materi untuk praktikum dikonfigurasi melalui dockerfile. Setiap materi praktikum memiliki konfigurasi yang berbeda sesuai kebutuhan materi pada bab yang akan dibuat praktikum. Ada tiga materi praktikum yang akan dirancang dalam penelitian ini. Berikut ini konfigurasi setiap *image* praktikum.

5.2.1.1 *Packet Capturing*

Bab pertama pada praktikum jaringan komputer ini adalah *packet capturing*. Untuk dapat melakukan *packet capturing* dibutuhkan beberapa komponen yang menunjang seperti berikut langkah-langkahnya.

Tabel 5.1 Konfigurasi Dockerfile Materi Packet Capturing

```
1  FROM ubuntu
2  RUN apt-get update
3  RUN mkdir /Capturing
4  RUN cd Capturing
5  RUN apt-get install tcpdump
```

Tabel 5.1 menunjukkan *script* yang dijalankan untuk melakukan instalasi tcpdump untuk *packet capturing*. Berikut penjelasan dari *script* diatas:

- Baris 1 mendefinisikan sebuah *base image* untuk membangun proses pada setiap *docker image* pada *repository ubuntu*
- Baris 2 merupakan perintah untuk melakukan *update packet-packet* yang tersedia pada *container* ubuntu.
- Baris 3 merupakan perintah untuk membuat sebuah *folder* dengan nama "Capturing".
- Baris 4 adalah perintah untuk masuk ke folder *capturing*.
- Baris 5 merupakan perintah untuk melakukan instalasi *tcpdump* sebagai perangkat lunak atau tools untuk praktikum packet capturing.

Setelah *dockerfile* dikonfigurasi sesuai kebutuhan praktikum, *dockerfile* di-*build* menggunakan perintar *docker build -t [Repository:tag] [path]*. Tujuan *build* adalah untuk membuat *dockerfile* menjadi *image*. Pada kasus ini Pada kasus ini penulis menggunakan Imandos/packet_capturing diamana Imandos adalah sebagai *user* dan *packet-capturing* adalah *repository* pada *dockerhub* dan *path*.

Proses selanjutnya adalah *push image* pada *docker registry*. Untuk melakukan proses *push* diperlukan akun pada *dockerhub* sebagai *repository* untuk menyimpan *image* secara *cloud*. *Dockerhub* memungkinkan *user* untuk membuat *repository* miliknya sendiri. *Push image* dapat dilakukan dengan perintah *docker push* [*user/repository:tag*]. *Docker push* merupakan perintah untuk melakukan *push* atau *upload* pada *repository* yang telah dibuat pada *dockerhub*. *Repository* merupakan nama yang sama dengan *repository* yang ada pada *dockerhub*. Pada kasus ini penulis menggunakan perintah Imandos/packet_capturing:1.0

5.2.1.2 Pemrograman *Socket*

Bab kedua pada jaringan komputer merupakan pemrograman *socket*, untuk melaukan praktikum ini dibutuhkan bebarapa *tools* seperti python, berikut ini merupakan *dockerfile* yang akan dijalankan :

Tabel 5.2 Konfigurasi *Dockerfile* Pemrograman *Socket* (*Server)*

```
1  FROM ubuntu
2  RUN apt-get update -y
3  RUN apt-get install -y python
4
5  RUN touch server.py
6  RUN echo "import socket" >> server.py
```

```
7   RUN echo "sock=socket.socket(socket.AF_INET,
    socket.SOCK_STREAM)" >> server.py
8   RUN echo "server=('',7777)" >> server.py
9   RUN echo "sock.bind(server)" >> server.py
10  RUN echo "sock.listen(1000)" >> server.py
11  RUN echo "print 'Server menerima koneksi'" >> server.py
12  RUN echo "conn, addr = sock.accept()" >> server.py
13  RUN echo "data1=conn.recv(1024)" >> server.py
14  RUN echo "data='OK ' +data1" >> server.py
15  RUN echo "conn.send(data)" >> server.py
16  RUN echo "print data" >> server.py
17  RUN echo "conn.close()" >> server.py
18
19  CMD python server.py
```

Tabel 5.3 Konfigurasi *Dockerfile* Pemrograman *Socket* (Client)

```
1   FROM ubuntu
2   RUN apt-get -y update
3   RUN apt-get install -y python
4
5   RUN touch client.py
6   RUN echo "import socket" >> client.py
7   RUN echo "sock=socket.socket(socket.AF_INET,
    socket.SOCK_STREAM)" >> client.py
8   RUN echo "server=('172.17.0.2',7777)" >> client.py
9   RUN echo "sock.connect(server)" >> client.py
10  RUN echo "masuk=raw_input('Input: ')" >> client.py
11  RUN echo "sock.send(masuk)" >> client.py
12  RUN echo "data1=sock.recv(1024)" >> client.py
13  RUN echo "print data1" >> client.py
14  RUN echo "sock.close()" >> client.py
15
16  CMD python client.py
```

Tabel 5.2 dan 5.3 menunjukkan *script* yang dijalankan untuk melakukan instalasi perangkat lunak *python* untuk praktikum permrograman *socket*. Berikut penjelasan dari *script* diatas:

- Baris 1 mendefinisikan sebuah *base image* untuk membangun proses pada setiap *docker image* pada *repository ubuntu*
- Baris 2 merupakan perintah untuk melakukan *update* packet-packet yang tersedia pada *container* ubuntu.
- Baris 3 merupakan perintah untuk melakukan instalasi aplikasi *python* sebagai perangkat lunak kebutuhan praktikum pemrograman *socket*.
- Baris 5 berfungsi untuk membuat *file* dengan nama server.py
- Baris 6 – 17 pada tabel 5.2 merupakan konfigurasi untuk membuat *server* pada *python* kemudian konfigurasi tersebut dimasukan ke dalam file server.py yang telah dibuat sebelumnya dengan perintah *echo*. Sedangkan baris 6-14 pada tabel 5.3 merupakan konfigurasi untuk membuat *client* pada *python*.

- Baris yang terakhir merupakan perintah untuk mengeksekusi konfigurasi *python* tersebut ketika dijalankan untuk praktikum.

Setelah *dockerfile* dikonfigurasi sesuai kebutuhan praktikum, *dockerfile* di-*build* menggunakan perintar *docker build -t [Repository:tag] [path]*. Tujuan *build* adalah untuk membuat *dockerfile* menjadi *image*. Pada kasus ini Pada kasus ini penulis menggunakan Imandos/pemsock_server dan Imandos/pemsock_client diamana Imandos adalah sebagai *user* dan pemsock_server juga pemsock_client adalah *repository* pada *dockerhub* dan *path*.

Proses selanjutnya adalah *push image* pada *docker registry*. Untuk melakukan proses *push* diperlukan akun pada *dockerhub* sebagai *repository* untuk menyimpan *image* secara *cloud*. *Dockerhub* memungkinkan user untuk membuat *repository* miliknya sendiri. *Push image* dapat dilakukan dengan perintah *docker push [user/repository:tag]. Docker push* merupakan perintah untuk melakukan *push* atau *upload* pada *repository* yang telah dibuat pada *dockerhub*. *Repository* merupakan nama yang sama dengan *repository* yang ada pada *dockerhub*. Penulis melakukan dua kali *push* karena ada dua *image* yaitu *server* dan *client*. Pada kasus ini menggunakan perintah docker push Imandos/pemsock_server:1.0 dan docker push Imandos/pemsock_client:1.0

5.2.1.3 *Protocol Transport Layer*

Pada bab selanjutnya praktikum jaringan komputer berisi tentang *protocol transport layer*. Untuk melakukan praktikum ini dibutuhkan perangkat lunak yang mendukung, oleh sebab itu akan disediakan *vsftpd* sebagai perangkat lunak pendukung untuk melakukan praktikum. Berikut ini merupakan *dockerfile* yang akan dijalankan :

Tabel 5.4 Konfigurasi Dockerfile Praktikum Protokol Layer transport

```
1   FROM ubuntu
2
3   RUN apt-get update && apt-get install -y --no-
    install-recommends vsftpd
4   RUN apt-get clean
5
6   RUN echo "local_enable=YES" >> /etc/vsftpd.conf
7   RUN echo "chroot_local_user=YES" >>
    /etc/vsftpd.conf
8   RUN echo "write_enable=YES" >> /etc/vsftpd.conf
9   RUN echo "local_umask=022" >> /etc/vsftpd.conf
10  RUN echo "anonymous_enable=NO" >> /etc/vsftpd.conf
11  RUN echo "allow_writeable_chroot=YES" >>
    /etc/vsftpd.conf
12  RUN echo "secure_chroot_dir=/home/user/files" >>
    /etc/vsftpd.conf
13
14  RUN useradd -ms /bin/bash user
15  RUN echo "user:imandos88" | chpasswd
16  RUN chmod 777 /home/user
```

```
17    RUN mkdir /home/user/files
18    RUN chown user:user /home/user/files/
19
20    VOLUME /etc
21
22    EXPOSE 20 21
23
24    CMD /usr/sbin/vsftpd
```

Tabel 5.4 menunjukkan *script* yang dijalankan untuk melakuakan instalasi *vsftpd* sebagai perangkat lunak pendukung untuk praktikum paket *capturing*. Berikut penjelasan dari *script* diatas:

- Baris 1 mendefinisikan sebuah *base image* untuk membangun proses pada setiap *docker image* pada *repository ubuntu*
- Baris 3 merupakan perintah untuk melakukan *update* pada *repository ubuntu* dan melakukan instalasi *vsftp* sebagai perangkat lunak untuk kebutuhan praktium.
- Baris 4 merupakan perintah untuk menghapus isi *cache* pada *disk* yang digunakan saat menjalankan perintah *apt-get install* secara otomatis dengan cara menghapus versi paling lama dari suatu *package*.
- Baris 6 - 12 merupakan perintah untuk melakukan konfigurasi pada *file* vsftpd.conf dengan konfigurasi yang telah dibuat sendiri untuk keperluan praktikum. Konfigurasi tersbut secara otomotasi akan me-*replace* konfigurasi sebelumnya atau konfigurasi *default* dari *vsftpd*.
- Baris 14 dan 15 merupakan perintah membuat *user* dan *password* baru untuk *login* pada *ftp*
- Baris 16-18 merupakan perintah untuk merubah hak akses dan hak kepemilikan pada *directory home/user.*
- Baris 20 perintah *volume* digunakan untuk mengaktifkan akses dari *container* kita ke direktori pada *host*.
- Baris 22 merukaan perintah untuk menghubungkan *port* 20 dan 21 untuk mengaktifan *network* antara proses yang berjalan di dalam *container* dan mesin *host*.

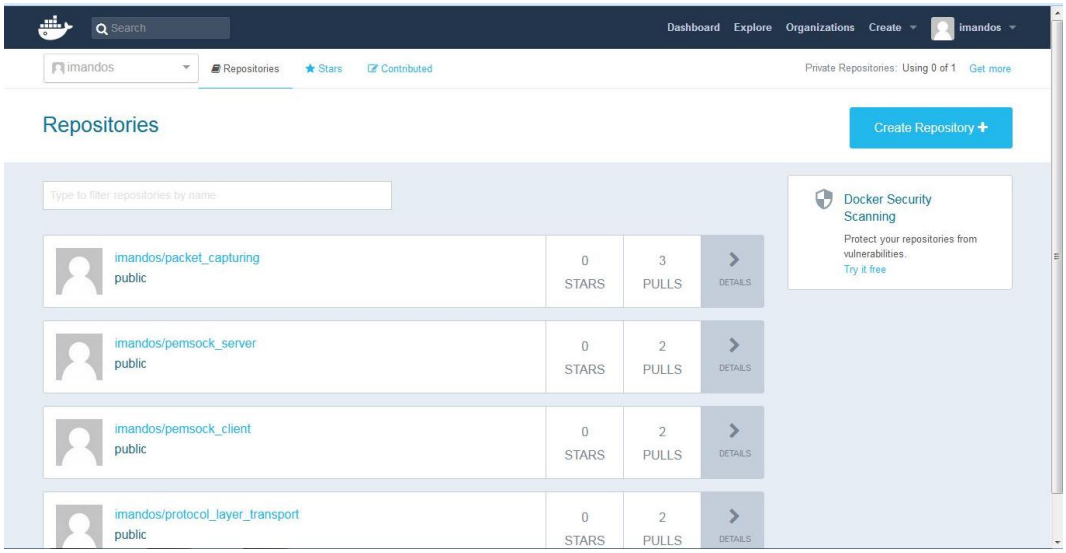
Setelah *dockerfile* dikonfigurasi sesuai kebutuhan praktikum, *dockerfile* di-*build* menggunakan perintar *docker build -t [Repository:tag] [path].* Tujuan *build* adalah untuk membuat *dockerfile* menjadi *image*. Pada kasus ini penulis menggunakan Imandos/protocol_layer_transport diamana Imandos adalah sebagai *user* dan *protocol_layer_transport* adalah *repository* pada *dockerhub* dan *path*.

Proses selanjutnya adalah *push image* pada *docker registry*. Untuk melakukan proses *push* diperlukan akun pada *dockerhub* sebagai *repository* untuk menyimpan *image* secara *cloud*. *Dockerhub* memungkinkan *user* untuk membuat *repository* miliknya sendiri. *Push image* dapat dilakukan dengan perintah *docker*

push [user/repository:tag]. Docker push merupakan perintah untuk melakukan *push* atau *upload* pada *repository* yang telah dibuat pada *dockerhub*. *Repository* merupakan nama yang sama dengan *repository* yang ada pada *dockerhub*. Pada kasus ini menggunakan perintah *docker push* Imandos/protocol_layer_ transport:1.0

5.2.2 Antarmuka Implementasi

Berikut ini merupakan tampilan atau gambar yang berkaitan dengan sistem yang dibuat dan di implementasikan pada tugas akhir ini.



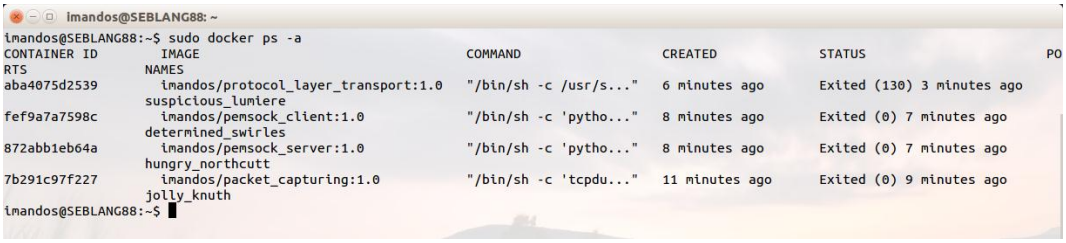
Gambar 5.1 *Repository Dockerhub*

Gambar 5.1 merupakan *repository* yang telah dibuat pada *dockerhub*. Terdapat tiga materi yang dibuat namu *repository* pada gambar ada empat yaitu packet capturing, pemrograman socket dan protocol layer transport. Permrograman socket memiliki dua *image* yaitu *server* dan *client*. keempat *repository* tersebut harus memiliki nama yang sama dengan *image* yang akan di push.

```
imandos@SEBLANG88: ~
imandos@SEBLANG88:~$ sudo docker images
[sudo] password for imandos:
REPOSITORY                      TAG                 IMAGE ID            CREATED             SIZE
imandos/packet_capturing        1.0                 664962933548        About an hour ago   165MB
imandos/pemsock_client          1.0                 47a28005fef1        About an hour ago   192MB
imandos/pemsock_server          1.0                 fe7e17f14591        About an hour ago   192MB
imandos/protocol_layer_transport   1.0              71775fd97d0f        About an hour ago   164MB
ubuntu                          latest              14f60031763d        3 months ago        120MB
imandos@SEBLANG88:~$
```

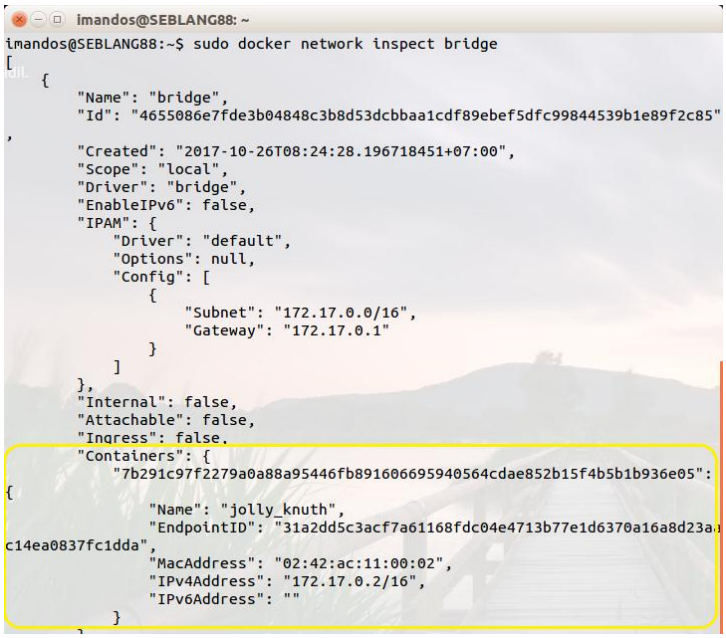
Gambar 5.2 Daftar *Docker Image* Yang dibuat

Gambar 5.2 merupakan daftar *image* yang telah dibuat, pada gambar terdapat tiga *image* yang masing-masing *image* di *generated* dari *dockerfile*. Setiap *imgae* memiliki image ID yang berbeda. *Tag* berfungsi sebagai penanda ataupun pembeda untuk setiap *image* dengan nama yang sama.



Gambar 5.3 Daftar *Container* Yang Telah Dibuat

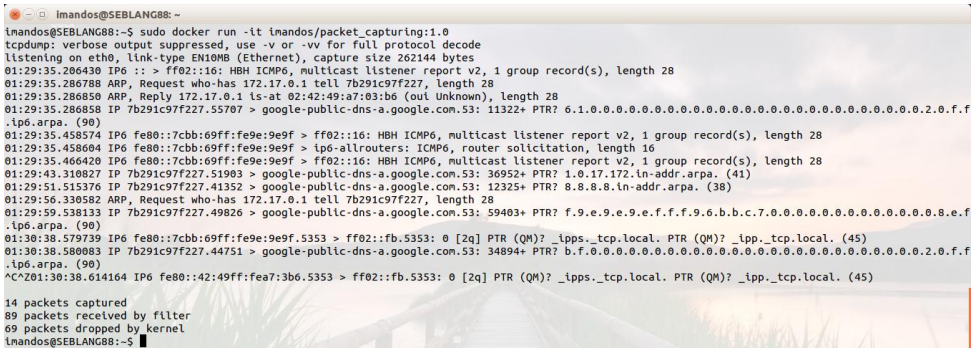
Gambar 5.3 adalah daftar semua *container* yang telah dibuat. dengan status *exited* karena sudah tidak digunakan. Saat container tersebut digunakan kembali untuk praktikum maka status akan berubah menjadi *up*.



Gambar 5.4 Koneksi *Bridge* Pada *Container*

Gambar 5.4 merupakan salah satu *container* yang terhubung internet menggunakan *bridge*. Setelah container dibuat secara otomatis ditambahkan pada koneksi *bridge* dan mendapatkan *ip addres* sesuai *subnet*.

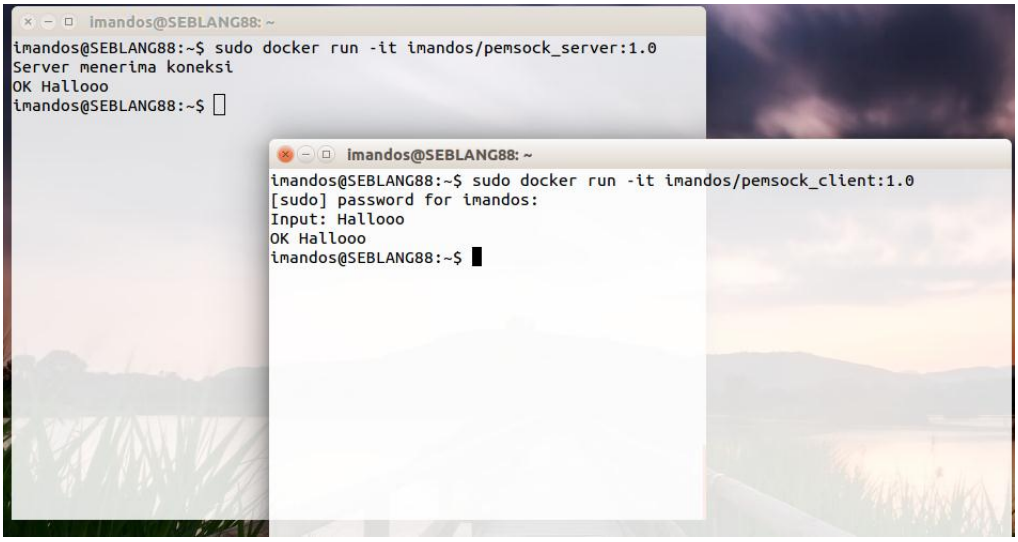
5.2.2.1 Praktikum Packet Capturing



Gambar 5.5 Packet Capturing

Pada gambar 5.5 dilakukan penangkapan *packet* terhadap jaringan *eth0*. Kemudian proses di hentikan dan mendapatkan hasil 14 packet captured, 89 pcket received by filter dan 69 packet *dropped by kernel*.

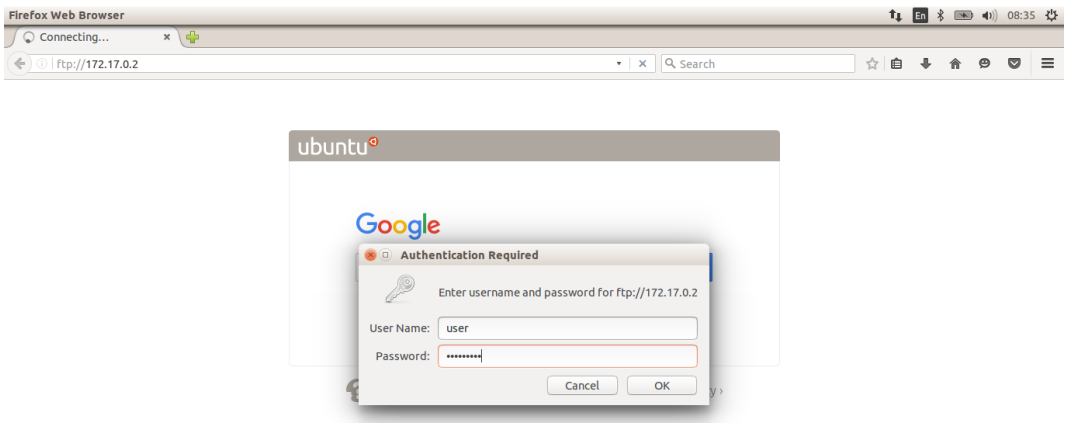
5.2.2.2 Praktikum Pemrograman Socket



Gambar 5.6 Pemrograman *Socket*

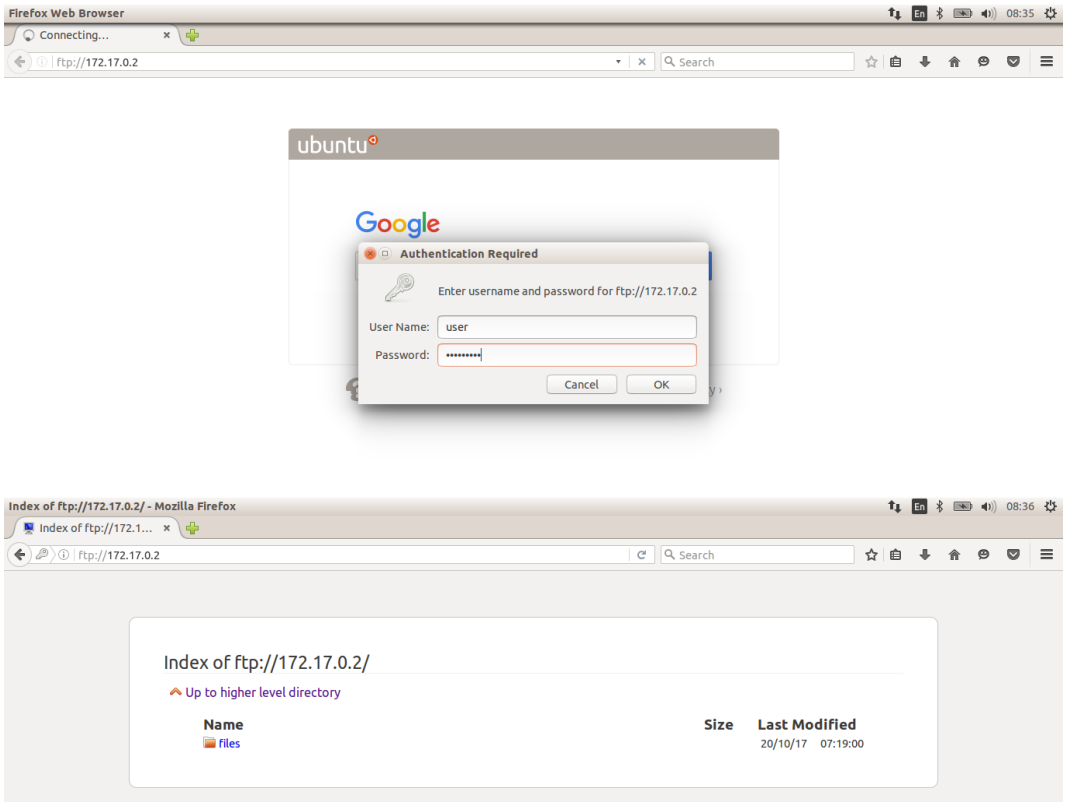
Gambar 5.6 merupakan praktikum pemrograman *socket* saat dijalankan. Dua *image* dijalankan untuk praktikum pemrograman *socket*, *image* satu sebagai *server* dan *image* yg kedua sebagai *client*. Saat *server* dijalankan maka *server* akan menampilkan *output* "server menerima koneksi" selanjutnya *image client* dijalankan dan *user* diminta *input* untuk di kirim ke *server*. *server* menerima dan membalas dengan pesan "OK input".

5.2.2.3 Praktikum Pemrograman *Socket*



Gambar 5.7 *Protocol Layer Transport*

Gambar 5.7 merupakan praktikum *protocol layer transport* saat dijalankan. *Image* yang dijalankan merupakan *ftp server*. untuk pengaksesan *ftp server* tersebut dapat menggunakan *browser* maupun *filezilla* untuk *transfer file*.



Gambar 5.8 Pengaksesan FTP Pada *Browser*

Gambar 5.8 adalah file FTP yang diakses melalui *browser* dengan alamat 172.17.0.2 selanjutnya *browser* akan meminta *user* dan *password*. Kemudian untuk *upload* file pada ftp dapat menggunakan *filezilla*.