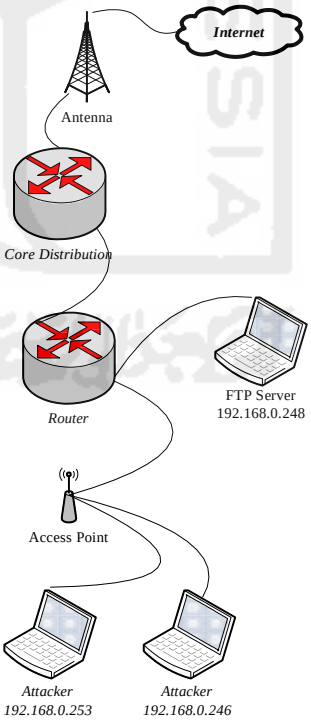
BAB III
METODOLOGI PENELITIAN

Dalam bagian ini dijelaskan aktifitas yang dilakukan dalam melakukan penelitian dibagi menjadi 2 (dua) yaitu: 1) Perancangan Skenario; dan 2) Penerapan Skenario.

3.1. Perancangan Skenario

3.1.1. Skenario Topologi Jaringan

Pada bagian ini menjelaskan skenario topologi jaringan yang digunakan dalam melakukan penelitian analisis *log* serangan *denial of service*, pendekatan yang digunakan menggunakan topologi jaringan hirarki (*tree*). Topologi ini membagi perangkat jaringan berdasarkan fungsinya dalam memberikan layanan. Skenario topologi jaringan yang digunakan ditunjukan pada **gambar 3.1.**



Gambar 3.1. Skenario Topologi Jaringan

Berikut ini adalah penjelasan masing-masing bagian dari topologi yang digunakan yaitu:

a. *Core Distribution*

Berfungsi sebagai *gateway* utama yang berinteraksi langsung dengan jaringan internet dan memberikan akses layanan internet bagi perangkat lain dalam Virtual LAN.

b. Router

Berfungsi sebagai *Gateway*, *router* ini tidak secara langsung berinteraksi dengan internet melainkan melalui *Core Distribution* untuk dapat mengakses internet.

c. Access Point

Berfungsi dalam menyediakan layanan internet menggunakan gelombang radio bagi perangkat pengguna.

d. *FTP Server* (Korban)

Bertujuan dalam mendukung aktifitas penelitian yang dilakukan. fungsi alat dijelaskan sebagai berikut:

- Berfungsi untuk melakukan aktifitas penelitian pada studi kasus yang digunakan.
- Berfungsi sebagai penyedia layanan transfer berkas (`FTP Server`) yang menggunakan *port* 21 sebagai saluran dalam melakukan transaksi.

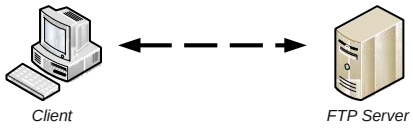
e. *Attacker* (Penyerang)

Berperan sebagai penyerang yang melakukan serangan *denial of service* pada layanan transfer berkas.

3.1.2. Skenario *Client-Server*

Pada bagian ini menjelaskan skenario *client-server*, pendekatan yang digunakan menggunakan arsitektur *client-server*, arsitektur ini digunakan dalam melakukan simulasi layanan transfer berkas antar

penyedia dan pengguna. Arsitektur ini akan menunjukan masing-masing peran perangkat pada jaringan. Skenario *client-server* yang digunakan ditunjukan pada **gambar 3.2.**



Gambar 3.2. Skenario Client-Server

Berikut ini adalah penjelasan masing-masing bagian dari topologi yang digunakan yaitu:

a. **`FTP Server`**

Berperan sebagai penyedia layanan transfer (*upload* & *download*) berkas.

b. ***Client***

Berperan sebagai pengguna layanan yang diberikan.

3.1.3. Skenario Serangan *Denial of Service* (*DoS*)

Pada bagian ini menjelaskan skenario *DoS*, jenis serangan digunakan adalah jenis serangan yang membebani jaringan dengan data sehingga penyedia layanan mengalami penurunan kinerja atau bahkan berhenti. Skenario ini akan menunjukan masing-masing peran dalam melakukan simulasi serangan. skenario *DoS* yang digunakan ditunjukan pada **gambar 3.3.** Berikut ini adalah penjelasan masing-masing bagian dari topologi yang digunakan yaitu:

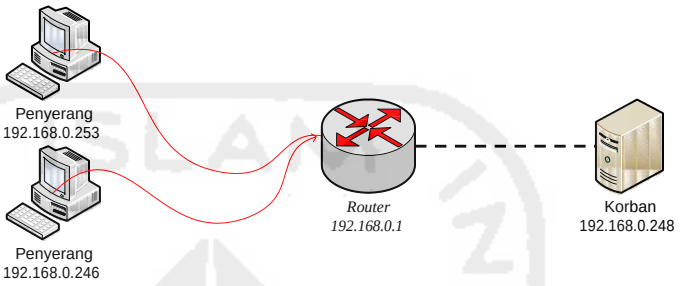
a. Penyerang (*attacker*)

Penyerang menggunakan komputer sebagai alat untuk membanjiri layanan (korban) dengan bantuan alat bantu LOIC dengan target *port* 21 dan FTP *Bruteforce* menggunakan *port* 443, serangan *flooding* dilakukan dengan menggunakan 2 (dua) buah komputer yang berperan sebagai penyerang dengan satu koneksi jaringan lokal pada korbannya, jenis serangan yang dilakukan berjenis serangan yang

membanjiri jaringan dengan banyak data (*traffic flooding*) pada FTP Server.

b. Korban (*victim*)

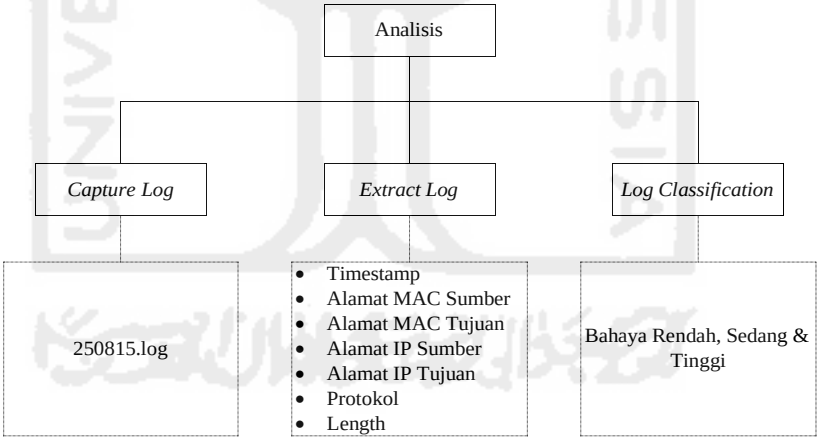
Korban adalah penyedia layanan transfer berkas.



Gambar 3.3. Serangan DoS

3.1.4. Skenario Analisis

Pada bagian ini menjelaskan skenario yang dikembangkan dalam melakukan penelitian, melibatkan 3 (tiga) pendekatan dalam. Skenario analisis ditunjukan pada **gambar 3.4**.



Gambar 3.4. Skenario Analisis

Berikut ini adalah penjelasan masing-masing tahapannya, yakni:

a. *Capture Log*

Berfungsi untuk mencatat aktifitas lalu lintas jaringan, alat yang digunakan dalam melakukan perekaman aktifitas menggunakan aplikasi `TCPDUMP`, perekaman ini dilakukan untuk mendapatkan informasi *timestamp*; alamat `MAC`; alamat `IP`; `port`; protokol dan

header. Hasil rekaman tersebut disimpan kedalam *log* berbentuk teks.

b. Log Extraction

Berfungsi untuk mengambil informasi dari berkas *log*, hasil dari proses ekstraksi ini kemudian disimpan pada basisdata. Alat bantu untuk melakukan ektraksi dan menyimpan dalam basisdata dari berkas *log* menggunakan skrip `regex`, skrip `regex` yang digunakan memiliki 2 (dua) fungsi yaitu: 1) mencari dan mencocokkan pola berdasarkan kriteria yang dibutuhkan seperti informasi *timestamp*; alamat `MAC`; alamat `IP`; alamat `port`; protokol dan *header*, dan 2) menyimpan informasi tersebut pada basisdata, alat bantu yang berfungsi sebagai media simpan hasil ekstraksi informasi dari skrip `regex` tersebut adalah `MySQL`.

c. Log Classification

Berfungsi untuk melakukan pengelompokkan basisdata *log*, tujuan dilakukan pengelompokkan tersebut adalah untuk mengetahui frekuensi ukuran paket data perjam. Dengan melakukan pengelompokkan data diharapkan pengetahuan akan apa yang terjadi dapat diketahui (dideteksi).

3.1.5. Skenario *Data Mining*

Pada bagian ini menjelaskan skenario dalam melakukan proses pencarian pengetahuan (KDD) dari sebuah basisdata *log* lalu lintas jaringan. Basisdata *log* didapatkan dari ekstraksi yang telah dilakukan dengan menggunakan skrip *regex* yang telah dilakukan sebelumnya, **gambar 3.5.** menunjukan skenario yang digunakan dalam melakukan proses pencarian pengetahuan (KDD) dari sebuah basisdata *log*. Berikut ini penjelasan masing-masing bagian dalam proses pencarian pengetahuan, yaitu:

Gambar 3.5. Skenario Ekstraksi Data Jaringan

a. Berkas *log* `250815.log`,

Merupakan berkas berjenis teks yang berisi lalu lintas data jaringan yang dilakukan selama 18 (delapan belas) hari. Berkas tersebut memiliki ukuran 1.5GB yang berisi + 11 juta *record* (setelah dimasukan pada basisdata).

b. Skrip `regex`

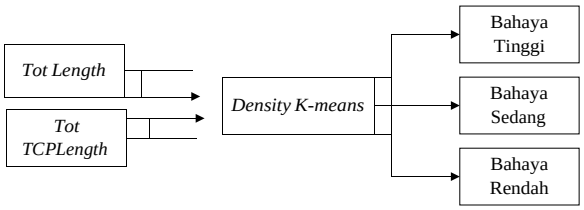
Merupakan alat bantu yang digunakan untuk mencari pola yang ditentukan, pola yang ditentukan dijelaskan diatas. kemudian menyimpannya dalam basisdata.

c. Basisdata *log*

Merupakan hasil akhir dari proses ekstraksi data jaringan. Data disimpan dalam satu basisdata yang memiliki satu tabel berlabel `paket_mikro`; dengan *field* yang telah dibuat sebelumnya sehingga skrip `regex` dapat langsung menyimpan pola yang ditemukan kedalam tabelnya. Data yang dibutuhkan dalam proses *clustering* adalah data yang tercatat pada tanggal 11 September 2015, pada tanggal tersebut telah dilakukan skenario serangan *DoS* pada skenario *file server*.

3.1.6. Skenario *Clustering*

Pada bagian ini menjelaskan skenario *clustering* yang dilakukan, pendekatan dalam melakukan *clustering* dengan menggunakan algoritma *Density K-means*). Skenario *clustering* ditunjukan pada **gambar 3.6.** Berikut ini adalah penjelasan dari masing-masing tahap dalam melakukan pengelompokkan basisdata *log*, yaitu:



Gambar 3.6. Skenario Clustering

a. Total *length* & Total TCP *length*

Kedua data tersebut didapatkan dari proses meringkas basisdata *log*. Berikut ini potongan basisdata *log* pada variabel yang digunakan adalah total *length* & total *TCPLength*. Kedua data tersebut dianggap sebagai data 2 (dua) dimensi, nilai yang dimiliki oleh kedua data tersebut akan dijadikan *input* dalam proses *clustering*.

b. Density K-means

Proses *clustering* dengan algoritma *K-means* ditunjukan pada **gambar 3.6.** berikut ini langkah-langkah pengelompokkan frekuensi data yang dikelompokan berdasarkan jam menggunakan algoritma *K-means.*

- Diketahui (x_{11}, x_{21}), (x_{12}, x_{22}), …, (x_{1n}, x_{2n}), dimana x_1 = frekuensi *length* perjam dan x_2 = frekuensi *tcplength* perjam.
- Inisiasi *k*, *k* = 3 (jumlah *cluster*) yang mewakili jenis kelompok bahaya rendah, bahaya sedang dan bahaya tinggi.
- Menentukan nilai terkecil (bahaya sedang), nilai tengah (bahaya sedang, nilai terbesar (bahaya tinggi), nilai-nilai tersebut digunakan sebagai inisiasi dalam menentukan *centroid* awal (C_1, C_2, C_3).
- Hitung jarak menggunakan **persamaan 2.1.** masing-masing data berdasarkan nilai *centroid*.

$$D(\chi,y)=\|\chi-y\|_2=\sqrt{\sum_{j=1}^{N}|x-y|^2}$$

- *Cluster*-kan *n* entitas sebagai berikut:

  Untuk *i* = 1 s/d *n* hitung:

$$c_1 = D(\chi_i, c_1) = \|\chi_i - C_1\|_2 = \sqrt{\sum_{i=1}^{N} |\chi_{ii} - c_{1i}|^2}$$

$$c_1 = D(\chi_i, c_2) = \|\chi_i - C_2\|_2 = \sqrt{\sum_{i=1}^{N} |\chi_{ii} - c_{2i}|^2}$$

$$c_1 = D(\chi_i, c_3) = \|\chi_i - C_3\|_2 = \sqrt{\sum_{i=1}^{N} |\chi_{ii} - c_{3i}|^2}$$

  Selanjutnya membandingkan nilai yang dimiliki oleh masing-masing data berdasarkan *centroid*-nya.

  $C_{min\text{-}i}$ = *min* (C_1, C_2, C_3)

- Hitung *Centroid* Baru

  $C_1(x_1, x_2)$ rerata dari C_1

  $C_2(x_1, x_2)$ rerata dari C_2

  $C_3(x_1, x_2)$ rerata dari C_3

$$\bar{\chi} = \frac{\chi_1 + \chi_2 + \ldots + \chi_n}{n}$$

- Jika $Centroid_{\text{Baru}}$ = $Centroid_{\text{Lama}}$ maka proses *clustering* dengan menggunakan algoritma *k-means* selesai, jika tidak, ulangi langkah 6.

Flowchart dan penjelasan masing-masing tahapan dalam melakukan pengelompokkan basisdata *log* dengan menggunakan algoritma *Density k-means* ditunjukan pada **gambar 3.7**, yaitu:

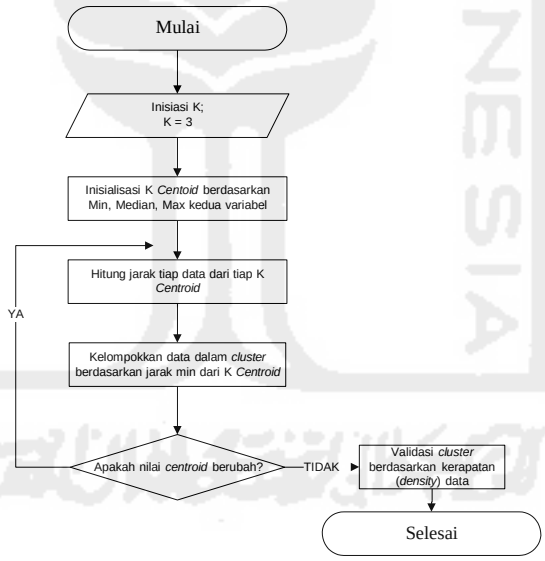
a. Melakukan inisiasi *k*, dimana *k* adalah jumlah *cluster* (kelompok) yang akan dibentuk.
b. Menentukan *centroid* awal yang diperoleh dari nilai terkecil, nilai tengah dan nilai terbesar dari variabel *length* dan *tcplength*.

c. Hitung jarak masing-masing data dengan masing-masing *centroid*.

d. Kelompokan data pada *cluster* berdasarkan jarak minimun dari *centroid*.

e. Bila nilai $centroid_{baru}$ dengan $centroid_{lama}$ tidak berubah, maka proses *clustering* dengan menggunakan algoritma *Density k-means* selesai, namun bila nilai $centroid_{baru}$ dengan $centroid_{lama}$ masih berubah, ulangi proses perhitungan jarak hingga nilainya tidak berubah.

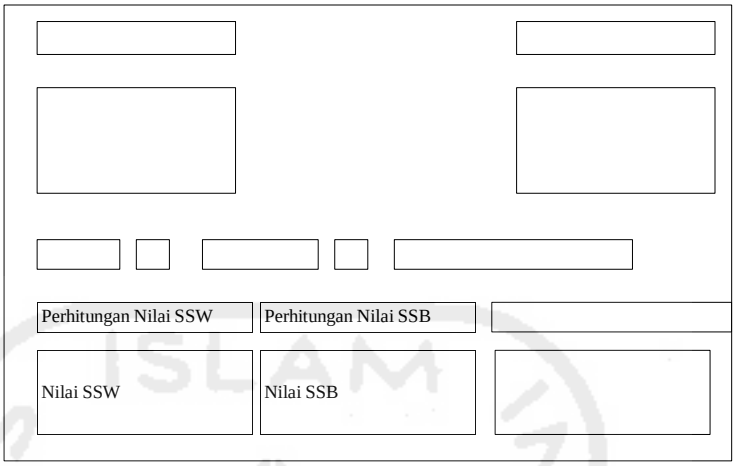
f. Hasil proses *clustering* iterasi terakhir kemudian divalidasi dengan menggunakan teknik validasi internal *Davies-Bouldin Index*.



Gambar 3.7. Flowchart Algoritma Density K-Means

3.1.7. Skenario Sistem

Pada bagian ini akan menjelaskan skenario perancangan sistem yang digunakan dalam melakukan *clustering* dengan algoritma *density k-means*. Skenario sistem ditunjukan pada **gambar 3.8.**



Gambar 3.8 Rancangan Sistem Density K-means

Ketika sistem dijalankan otomatis akan membaca data *totlength* dan *tottcplength* dan kemudian menampilkan grafik awal sebelum dilakukan proses *clustering*, setelahnya baru memasukan jumlah *cluster* yang ditentukan dalam hal ini jumlah *cluster* telah ditentukan sebanyak 3 (tiga), dan memberikan nilai proses *clustering* dilakukan hingga menemui kondisi data sudah tidak lagi berpindah antar *cluster* (konvergen), dalam hal ini jumlah iterasi ditentukan sebanyak 4 (empat) kali, setelah kedua nilai diberikan kemudian dijalankan proses *clustering* yang kemudian sistem akan menampilkan hasil validasi *cluster* berdasarkan tingkat kerapatan data pada pusatnya, nilai validasi *cluster* berupa hasil perhitungan nilai *SSW* (*Sum of Square Within Cluster*) atau nilai kohesi (*density*) data pada pusatnya, hasil perhitungan nilai *SSB* (*Sum of Square Between Cluster*) atau nilai separasi (keterpisahan) dan terakhir adalah nilai *R* dan *DBI*.

3.2. Penerapan Skenario

3.2.1. Penerapan *Log Capturing*

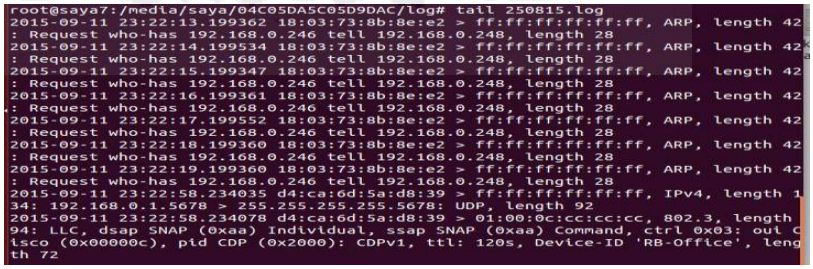
Pada bagian ini merupakan awal dalam menerapkan *log capturing*, pengumpulan informasi yang didapatkan pada penelitian, menitik beratkan pada lalu lintas data jaringan LAN Mandala Citra Media di Surakarta dengan arsitektur yang ditunjukan pada **gambar 3.1.** Pengumpulan data lalu lintas jaringan dilakukan pada tanggal 25

Agustus 2015 sampai dengan 11 September 2015 atau sama dengan 18 (delapan belas hari) dengan melakukan pemantauan dan pencatatan aktifitas jaringan LAN. Alat bantu yang digunakan untuk melakukan perekaman lalu lintas data adalah *packet sniffer* `TCPDUMP` yang tersedia pada kebanyakan sistem operasi linux atau dapat diperoleh di `www.tcpdump.org`, pemilihan alat bantu ini disebabkan oleh karakteristiknya yang mampu merekam lalu lintas data jaringan komputer berdasarkan kriteria pemakai dalam berbagai format yang diharapkan. Perintah untuk melakukan pengumpulan lalu lintas data jaringan komputer ditunjukan pada **gambar 3.9.**

```
tcpdump –i eth0 –tttt –n –q –e > 250815.log
```

Gambar 3.9. Potongan perintah `TCPDUMP`

Perangkat lunak tersebut berfungsi untuk melakukan proses pengumpulan pada semua lalu lintas data yang melewati *interface* `eth0` dan menampilkan *outputnya* berdasarkan *timestamp*, format alamat IP dan menampilkan informasi protokol dan *header* dari paket data tersebut. Hasilnya berupa berkas berjenis teks dengan nama `250815.log`. *log* ini kemudian disimpan untuk digunakan sebagai *log* asli untuk melakukan verifikasi jika diperlukan dalam penyelidikan forensik sesungguhnya. Potongan berkas *log* ditunjukan pada **gambar 3.10.**



Gambar 3.10 Potongan Berkas Log

3.2.2. Penerapan *Log Extraction*

Pada bagian ini menjelaskan langkah kedua dalam menerapkan *log extraction*, yaitu melakukan ekstraksi data dari berkas *log* `250815.log`. Skrip lengkap untuk melakukan ekstraksi data dari Berkas *log* ditunjukan oleh **gambar 3.11.** Penjelasan dari masing-masing blok perintah adalah sebagai berikut:

- Baris 1, menjelaskan jenis *shell* yang digunakan dalam melakukan adalah *shell bash*.
- Baris 2 hingga Baris 5, menjelaskan variabel untuk melakukan hubungan dengan basisdata, basisdata yang digunakan pada penelitian ini menggunakan *Database Management System* (DBMS) `MySQL`.

```
1  #!/bin/bash
2  host=localhost
3  user=root
4  pass=qwerty
5  db=mikro
6  #ifconfig eth0 up
7  cat 250815.log | grep IPv4 | awk -F "[:,> ]"
8  {'print $1 " "$2":"$3":"$4" "$5":"$6":"$7":"$8":"$9":"$10"
9  "$13":"$14":"$15":"$16":"$17":"$18" "$25" "$28" "$30" "$23" "$31'}
10 | awk -F "[. ]" '{ if ($25=="")
11 print "INSERT IGNORE INTO paket_mikro
12 (`timestamps`, `mac_add_sbr`, `mac_add_tuj`, `ip_port_add_sbr`, `ip_port_add_tuj`,
13 protokol`, `length`, `tcplength`)
14 values
15 (\""$1" "$2"\", \""$4"\", \""$5"\", \""$6" "$7" "$8" "$9" "$10"\",
16 \""$11" "12" "$13" "$14" "$15"\", \""$16"\", \""$17"\", \""$18"\");"}'
17 | mysql -u$user -p$pass $db
```

Gambar 3.11 Skrip untuk Ekstraksi Data

- Baris 7 hingga Baris 8, menjelaskan proses untuk menampilkan data yang dicari untuk kemudian diekstrak, *output* dari baris tersebut menampilkan informasi *timestamps*, alamat *mac*, alamat *ip* dan *port*, protokol, ukuran paket.
- Baris 10 hingga Baris 11, merupakan lanjutan proses yang diawal menampilkan kriteria informasi yang diinginkan untuk selanjutnya disimpan dalam basisdata dengan menggunakan 1 (satu) tabel dengan *field* yang telah dibuat sebelumnya. Tabel yang telah dipersiapkan terlebih dahulu ditunjukan pada **gambar 3.12.**

Fields | Indexes | Foreign Keys | Triggers | Options | Comment | SQL Preview

Name	Type	Length
id	int	8
timestamps	timestamp	0
mac_add_sbr	varchar	20
mac_add_tuj	varchar	20
ip_port_add_sbr	varchar	20
ip_port_add_tuj	varchar	20
protokol	text	0
length	int	8
tcplength	int	8

Gambar 3.12. Struktur Tabel

Hasil ekstraksi yang telah dilakukan dengan menggunakan skrip tersebut dengan sumber data berasal dari berkas berjenis teks sebelumnya ditunjukan **gambar 3.13a** & **gambar 3.13b.**

id	timestamps	mac_add_sbr	mac_add_tuj
3606177	2015-09-10 23:31:	18:03:73:8b:8e:e2	b8:76:3f:a5:3c:bb
3606178	2015-09-10 23:31:	b8:76:3f:a5:3c:bb	18:03:73:8b:8e:e2
3606179	2015-09-10 23:31:	18:03:73:8b:8e:e2	b8:76:3f:a5:3c:bb
3606180	2015-09-10 23:31:	b8:76:3f:a5:3c:bb	18:03:73:8b:8e:e2
3606181	2015-09-10 23:31:	18:03:73:8b:8e:e2	b8:76:3f:a5:3c:bb
3606182	2015-09-10 23:31:	b8:76:3f:a5:3c:bb	18:03:73:8b:8e:e2
3606183	2015-09-10 23:31:	18:03:73:8b:8e:e2	b8:76:3f:a5:3c:bb
3606184	2015-09-10 23:31:	b8:76:3f:a5:3c:bb	18:03:73:8b:8e:e2
3606185	2015-09-10 23:31:	18:03:73:8b:8e:e2	b8:76:3f:a5:3c:bb
3606186	2015-09-10 23:31:	b8:76:3f:a5:3c:bb	18:03:73:8b:8e:e2
3606187	2015-09-10 23:31:	18:03:73:8b:8e:e2	b8:76:3f:a5:3c:bb
3606188	2015-09-10 23:31:	b8:76:3f:a5:3c:bb	18:03:73:8b:8e:e2
3606189	2015-09-10 23:31:	18:03:73:8b:8e:e2	b8:76:3f:a5:3c:bb

Gambar 3.13a. Basisdata log

Pada **gambar 3.14a**, menunjukkan informasi ***waktu*** dan ***tanggal***, ***mac_add_sbr*** dan ***mac_add_tuj*** adalah identitas komputer yang melakukan permintaan dan menerima permintaan. Pada **gambar 3.14b** menunjukan ***ip_port_add_sbr*** dan ***ip_port_add_tuj*** yang menunjukan alamat *ip* dan *port* asal dan tujuan komputer, protokol merupakan jalur yang digunakan dalam melakukan komunikasi, *length* dan *tcplength* merupakan ukuran paket yang ditransmisikan melalui protokol yang digunakan.

ip_port_add_sbr	ip_port_add_tuj	protokol	length	tcplength
192 168 0 248 21	192 12 0 253 37667	tcp	54	0
192 168 0 253 37667	192 12 0 248 21	tcp	630	576
192 168 0 248 21	192 12 0 253 37667	tcp	54	0
192 168 0 253 37667	192 12 0 248 21	tcp	438	384
192 168 0 248 21	192 12 0 253 37667	tcp	54	0
192 168 0 253 37667	192 12 0 248 21	tcp	1514	1460
192 168 0 248 21	192 12 0 253 37667	tcp	54	0
192 168 0 253 37667	192 12 0 248 21	tcp	66	12
192 168 0 248 21	192 12 0 253 37667	tcp	54	0
192 168 0 253 37667	192 12 0 248 21	tcp	1514	1460
192 168 0 248 21	192 12 0 253 37667	tcp	54	0
192 168 0 253 37667	192 12 0 248 21	tcp	66	12
192 168 0 248 21	192 12 0 253 37667	tcp	54	0

Gambar 3.13b. Basisdata log

3.2.3. Penerapan Ekstraksi Fitur

Pada bagian ini menjelaskan langkah ketiga dalam menerapkan ekstraksi fitur yang dibutuhkan sebagai input algoritma *density k-means*. Berkas *log* yang telah disimpan pada basisdata *log*, memiliki ukuran ±1.5GB, dengan total *records* sebanyak 11.358.001 *records*. Fitur yang dibutuhkan untuk melakukan *clustering* adalah frekuensi kemunculan data dengan atribut *length* & *tcplength*. Dalam menentukan tingkat bahaya dalam hal ini tingkat bahaya dikelompokkan menjadi bahaya rendah, sedang & tinggi dengan kriteria ukuran paket lebih dari 0, kurang atau lebih dari 10 dan kurang 100 dikategorikan sebagai bahaya tingkat rendah, kemudian untuk kriteria ukuran paket lebih dari 100 dan atau kurang dari 1000 dikategorikan sebagai bahaya tingkat sedang dan kriteria ukuran paket lebih dari 1000 dikategorikan sebagai bahaya tingkat tinggi. Dalam menerapkan kriteria tersebut digunakan perintah `mysql` yang ditunjukan pada **gambar 3.14a** & **gambar 3.14b**.

```
select hour(timestamps), length, count(length) from paket_mikro where length > 0 AND length < 10 group by hour(timestamps);
select hour(timestamps), length, count(length) from paket_mikro where length > 10 AND length < 100 group by hour(timestamps);
select hour(timestamps), length, count(length) from paket_mikro where length > 100 and length < 1000 group by hour(timestamps);
select hour(timestamps), length, count(length) from paket_mikro where length > 1000 group by hour(timestamps);
```

Gambar 3.14a. Ekstraksi Fitur Frekuensi length

```
select hour(timestamps), tcplength, count(tcplength) from paket_mikro where tcplength > 0 and tcplength < 10 group by hour(timestamps);
select hour(timestamps), tcplength, count(tcplength) from paket_mikro where tcplength > 10 and tcplength < 100 group by hour(timestamps);
select hour(timestamps), tcplength, count(tcplength) from paket_mikro where tcplength > 100 and tcplength < 100 group by hour(timestamps);
select hour(timestamps), tcplength, count(tcplength) from paket_mikro where tcplength > 1000 group by hour(timestamps);
```

Gambar 3.14b. Ekstraksi Fitur Frekuensi tcplength

hasil dari menjalankan perintah tersebut ditunjukan pada **tabel. 3.1a** & **tabel 3.1b**.

Tabel 3.1a length

Jam	> 0		> 10		> 100		> 1000	
	Length	Jml	Length	Jml	Length	Jml	Length	Jml
0	0	0	87	1237	134	2145	1122	22
1	0	0	87	843	134	1776	0	0
2	0	0	87	2151	134	1701	1030	10
3	0	0	87	1149	134	1765	1032	34
4	0	0	87	1962	134	1891	1118	18
5	0	0	87	2474	243	2124	1119	24
6	0	0	87	1792	347	1371	0	0
7	0	0	87	1497	243	1708	1514	48
8	0	0	87	1363	134	1548	0	0
9	0	0	64	1163	134	1485	0	0
10	0	0	87	1443	134	1907	1414	26
11	0	0	87	1130	134	1558	1119	4
12	0	0	87	1083	134	1925	1120	12
13	0	0	87	1525	134	2111	1122	22
14	0	0	87	3885	134	3707	1119	2217
15	0	0	87	16273	134	5787	1414	4811

Sambungan…

16	0	0	87	652760	134	126474	1122	204379
17	0	0	87	1793	134	2157	1484	23
18	0	0	87	1121	134	2577	1484	15
19	0	0	87	974	143	3123	1122	17
20	0	0	87	1155	143	2857	1484	17
21	0	0	87	1124	134	2267	1484	25
22	0	0	87	69374	134	8322	1514	82626
23	0	0	92	5576505	134	666708	1514	3861682

Setelah kriteria tingkat bahaya dari *length* dan *tcplength* didapatkan, kemudian menjumlahkan total frekuensi, contoh pada jam 0, didapatkan frekuensi sebanyak 1237, 2145, 22. Nilai yang didapatkan dari penjumlahan frekuensi tersebut sebanyak 3404, langkah yang sama juga diterapkan pada tabulasi *tcplength*. Hasil perhitungan lengkap total kedua frekuensi ditunjukan pada **tabel 3.2.**

Tabel 3.1b. TCPLength

Jam	> 0		> 10		> 100		> 1000	
	TCP	Jml	TCP	Jml	TCP	Jml	TCP	Jml
0	0	0	25	2	0	0	0	0
1	0	0	16	1	0	0	0	0
2	0	0	16	1	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	35	1	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1448	2
8	0	0	16	1	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	31	2	0	0	1348	10
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	7	49	16	464	0	0	1418	2175
15	8	204	75	1962	0	0	1348	4739
16	8	151	16	156461	0	0	1448	198910
17	6	16	29	61	0	0	1418	23
18	0	0	41	69	0	0	1418	15
19	0	0	35	77	0	0	1418	13
20	0	0	31	52	0	0	1418	17
21	0	0	65	59	0	0	1418	25

Sambungan…

22	2	337	16	3036	0	0	1448	82598
23	6	715	41	1142391	0	0	1460	3828411

Setelah dilakukan perhitungan pada *length* dan *tcplength* yang hasilnya berupa frekuensi kedua paket, ditunjukan pada **tabel 3.2.** kemudian menentukan nilai terkecil, tengah dan terbesar sebagai *threshold* (*centroid*) dalam mengklasifikasikan tingkat bahaya rendah, sedang dan tinggi, *threshold* nilai terkecil (rendah) untuk *length* sebanyak 2619, *tcplength* sebanyak 0; *threshold* nilai tengah (sedang) untuk *length* sebanyak 3685.5, *tcplength* 2, dan *threshold* nilai terbesar (tinggi) untuk *length* sebanyak 10104895, *tcplength* sebanyak 4971517.

Tabel 3.2. Frekuensi Length & TCPLength

Jam ke- ***i***	***TotLength***	***TotTCP***
0	3404	2
1	2619	1
2	3862	1
3	2948	0
4	3871	1
5	4622	0
6	3163	0
7	3253	2
8	2911	1
9	2648	0
10	3376	12
11	2692	0
12	3020	0
13	3658	0
14	9809	2688
15	26871	6905
16	983613	355522
17	3973	100
18	3713	84
19	4114	90
20	4029	69
21	3416	84
22	160322	85971
23	10104895	4971517

3.2.4. Penerapan *Clustering*

Pada bagian ini menjelaskan proses *clustering* berdasarkan jumlah frekuensi yang ditunjukan pada **tabel 3.2.** frekuensi *totlength* dan *tottcp* diperlakukan sebagai data 2 (dua) dimensi dalam proses *clustering* dengan algoritma *Density k-means*, adapun proses *clustering* dengan algoritma *density k-means* dijelaskan menjadi 6 (enam) tahapan yaitu:

1. Tentukan nilai *k* sebagai jumlah *cluster* yang ingin dibentuk.
2. Hitung *k centroid* (titik pusat *cluster*) awal berdasarkan nilai terkecil (*min*), nilai tengah (*median*) dan nilai terbesar (*max*) dari kedua fitur *totlength* & *tcplength.*
3. Hitung jarak setiap data ke masing-masing *centroid*
4. Setiap data memilih *centroid* yang terdekat.
5. Tentukan posisi *centroid* baru dengan cara menghitung nilai rata-rata dari data-data yang memilih pada *centroid* yang sama.
6. Ulang ke langkah 3 (tiga) jika data masih berubah.

Proses *clustering* menggunakan algoritma *Density K-means* dijelaskan sebagai berikut:

1. Langkah pertama menentukan jumlah *cluster*. Pada penelitian ini telah ditentukan banyaknya *cluster* sebanyak 3 (tiga) kelompok, yang mewakili kelas bahaya rendah, sedang dan tinggi.
2. Langkah kedua inisialisasi *k centroid*

   Untuk menentukan inisialisasi posisi awal *centroid* untuk masing-masing *cluster* dilakukan dengan menggunakan nilai terkecil (mewakili tingkat bahaya rendah), nilai tengah (tingkat bahaya sedang), dan nilai terbesar (tingkat bahaya tinggi). Contoh jam ke -1 dengan totlength dengan nilai terkecil = 2619; tengah = 3685.5 dan terbesar = 10104895, tottcplength dengan nilai terkecil = 0, tengah = 2 dan terbesar = 4971517, Ketiga nilai dari kedua atribut tersebut digunakan sebagai inisialisasi *centroid*.

3. Langkah ketiga perhitungan jarak

   Proses perhitungan jarak untuk jam ke-*1* ke *centroid* C_1 menggunakan **persamaan 2.1.**

$$D(\chi,y)=\|\chi-y\|_2=\sqrt{\sum_{j=1}^{N}|x-y|^2} \quad \text{.....} \qquad (2.1)$$

   - Jam ke-*1* ke *centroid* C_1 (Bahaya Rendah)

$$=\sqrt{(totlength-\min length)^2+(tottcpleng\,th-\min tcplength)^2}$$
$$=\sqrt{(3404-2619.00)^2+(2-0)^2}$$
$$=\sqrt{(785)^2+(2)^2}$$
$$=\sqrt{616225+4}$$
$$=\sqrt{616229}$$
$$=785$$

   - Jam ke-*1* ke *centroid* C_2 (Bahaya Sedang)

$$=\sqrt{(totlength-medianlength)^2+(tottcpleng\,th-mediantcplength)^2}$$
$$=\sqrt{(3404-3685.50)^2+(2-2)^2}$$
$$=\sqrt{(281)^2+(0)^2}$$
$$=\sqrt{78961+0}$$
$$=\sqrt{78961}$$
$$=281$$

   - Jam ke-*1* ke *centroid C*3 (Bahaya Tinggi)

$$=\sqrt{(totlength-\max length)^2+(tottcpleng\,th-\max tcplength)^2}$$
$$=\sqrt{(3404-10104895)^2+(2-4971517)^2}$$
$$=\sqrt{(10101491)^2+(4971515)^2}$$
$$=\sqrt{102040120423081+24715961395225}$$
$$=\sqrt{126756081818306}$$
$$=11258600.3$$

   perhitungan jarak ditunjukan pada **tabel 3.3.** Jam ke-*1* berjarak 785.00 dengan C_1, berjarak 281.50 dengan C_2 dan berjarak

11258600.35 dengan C_3, begitu juga dengan jam ke-*2* dan seterusnya. Setelah didapatkan perhitungan jarak lengkap, langkah berikutnya adalah menentukan *centroid* baru dengan cara menghitung nilai rata-rata dari data yang memilih pada *centroid* yang sama seperti yang dijelaskan pada poin nomor 4 (empat).

4. Langkah keempat menghitung *centroid* baru dengan cara menghitung nilai rata-rata dari data-data yang memilih pada *centroid* yang sama dengan menggunakan rumus rata-rata.

Tabel 3.3. Perhitungan jarak iterasi ke-1

Jam ke-*i*	C_1		C_2		C_3	
	2619.00	**0.00**	**3685.50**	**2.00**	**10104895.00**	**4971517.00**
0	785.00		281.50		11258600.35	
1	1.00		1066.50		11259305.12	
2	1243.00		176.50		11258189.86	
3	329.00		737.50		11259010.37	
4	1252.00		185.50		11258181.79	
5	2003.00		936.50		11257508.43	
6	544.00		522.50		11258817.46	
7	634.00		432.50		11258735.83	
8	292.00		774.50		11259043.12	
9	29.00		1037.50		11259279.54	
10	757.10		309.66		11258621.06	
11	73.00		993.50		11259240.06	
12	401.00		665.50		11258945.77	
13	1039.00		27.57		11258373.34	
14	7676.03		6686.69		11251667.56	
15	25215.84		24191.30		11234497.77	
16	1043429.50		1042426.20		10222778.25	
17	1357.69		303.74		11258046.56	
18	1097.22		86.49		11258286.90	
19	1497.71		437.44		11257924.46	
20	1411.69		349.97		11258010.00	
21	801.41		281.70		11258553.37	
22	179614.17		178677.54		11079850.72	
23	11259305.56		11258347.78		0.00	

Proses perhitungan nilai rata-rata dari data-data yang memilih pada *centroid* yang sama dijelaskan sebagai berikut:

$$Mean = \sum_{i=1}^{n} \frac{\chi_i}{n}$$

Keterangan:

x_i = data yang memilih *centroid* yang sama

n = banyak data pada *centroid* C_1

- *Centroid* Baru C_1 *totlength*

$$= \frac{2619 + 3862 + 2948 + 3871 + 4622 + 3163 + 3253 + 2911 + 2648 + 3376 + 2692 + 3020}{6}$$

$$= \frac{16838}{6}$$

$$= 2806.33333$$

- *Centroid* Baru C_1 *tottcplength*

$$= \frac{1+0+1+0+0+0}{6}$$

$$= \frac{2}{6}$$

$$= 0.33$$

Perhitungan rata-rata lengkap untuk iterasi pertama sampai dengan iterasi keempat (terakhir) ditunjukan pada **tabel 3.4**.

Tabel 3.4 Rata-rata pada Centroid yang sama

Iterasi	C_1		C_2		C_3	
1	2619.00	0.00	3685.50	2.00	10104895.00	4971517.00
2	2806.33	0.33	72298.18	26560.65	10104895.00	4971517.00
3	4.856	4.856	493.515	220.747	10.104.895	4.971.517
4	11922.45	4364.14	983613.00	355522.00	10104895.00	4971517.00

Pada iterasi keempat sudah tidak ditemukan adanya perpindahan data diantara *cluster*, sehingga proses iterasi pada *clustering* menggunakan algoritma *density k-means* dihentikan. Berdasarkan perhitungan rata-rata pada *centroid* yang sama menunjukan lokasi *centroid* menunjukan sudah tidak terjadi lagi perpindahan data diantara *cluster*. **Tabel 3.5** menunjukan lokasi *centroid* yang sudah tetap (konvergen).

Tabel 3.5 Jumlah data pada centroid

Iterasi	C_1	C_2	C_3	**Total**
1	6	17	1	24
2	21	2	1	24
3	22	1	1	24
4	22	1	1	24

Setelah didapatkan kondisi yang tetap pada *cluster*, kemudian proses pemberian label untuk tingkat bahaya rendah, sedang dan tinggi dilakukan. pelabelan hasil *clustering* dijelaskan lebih lengkap pada bab berikutnya.

3.2.5. *Davies-Bouldin Index*

Pada bagian ini akan menjelaskan proses validasi internal pada *cluster* berdasarkan tingkat densitas data dengan *centroid*, keterpisahan antar *cluster* dan rasio antar *cluster*, hasil perhitungan ketiga nilai tersebut dijadikan dasar untuk mendapatkan *Index Davies-Bouldin*. Untuk mendapatkan nilai *DBI* dilakukan melalui 3 (tiga) tahap yaitu: a) hitung *SSW*; b) hitung *SSB*; c) hitung nilai *R* & *DBI*.

Index Davies-Bouldin bertujuan untuk memaksimalkan (*separate*) jarak antar *cluster* & meminimumkan jarak antar titik dalam suatu *cluster* (*dense*). Nilai *index davies-bouldin* berada pada interval (0, 1), nilai minimum dari *index davies-bouldin* akan menunjukan jumlah *cluster* optimal.

Langkah pertama mengelompokkan data berdasarkan *cluster* yang diikuti, hal ini dilakukan agar dapat mengetahui data tersebut mengikuti *cluster* ke berapa, **tabel 3.6**. menunjukan data yang telah mengikuti *cluster* tertentu. Pada tabel tersebut menunjukan data paling banyak mengikuti *cluster* ke-1 yang merupakan jenis kelompok penggunaan normal (wajar), sedangkankan pada data yang mengikuti *cluster* 2 & 3 merupakan jenis kelompok data yang dicurigai sebagai serangan *DoS*.

Tabel 3.6. Cluster yang diikuti

Jam ke- *i*	**TotLength**	**TotTCP**	***Cluster* yg diikuti**
0	3404	2	1
1	2619	1	1
2	3862	1	1
3	2948	0	1
4	3871	1	1
5	4622	0	1
6	3163	0	1
7	3253	2	1
8	2911	1	1
9	2648	0	1
10	3376	12	1
11	2692	0	1
12	3020	0	1
13	3658	0	1
14	9809	2688	1
15	26871	6905	1
16	983613	355522	2
17	3973	100	1
18	3713	84	1
19	4114	90	1
20	4029	69	1
21	3416	84	1
22	160322	85971	1
23	10104895	4971517	3

Dari **tabel 3.6** tersebut kemudian dilakukan perhitungan rata-rata berdasarkan *centroid*-nya. C_1 memiliki anggota sebanyak 22 data dengan rata-rata *totlength* & *tottcplength* bernilai 11922.45455, 4364.136364, hasil perhitungan anggota pada tiap-tiap *cluster* ditunjukan oleh **tabel 3.7**. Banyak anggota pada *centroid*. C_2 anggota sebanyak 1 data dengan rata- *totlength* & *tottcplength* bernilai 983613, 355522, dan C_3 memiliki anggota sebanyak 1 data dengan rata-rata *totlength* & *tottcplength* bernilai 10104895, 4971517. Nilai rata-rata yang dimiliki oleh masing-masing C_1, C_2, dan C_3 dijadikan sebagai input dalam melakukan perhitungan kerapatan (*density*) data pada *cluster* atau *Sum of Square Within Cluster* (*SSW*). Nilai *centroid* yang dijadikan sebagai nilai dalam melakukan tingkat kerapatan data pada

suatu *cluster* ditunjukan oleh **tabel 3.8.** Hasil perhitungan rata-rata pada C_1, C_2, C_3 dijadikan sebagai nilai *centroid* yang ditunjukan pada **tabel 3.8**.

Tabel 3.7. Data pada C_1, C_2, C_3

Data pada C_1

totlength	*tottcplength*
3404	2
2619	1
3862	1
2948	0
3871	1
4622	0
3163	0
3253	2
2911	1
2648	0
3376	12
2692	0
3020	0
3658	0
9809	2688
26871	6905
3973	100
3713	84
4114	90
4029	69
3416	84
160322	85971

Data pada C_2

totlength	*tottcplength*
983613	355522

Data pada C_3

totlength	*tottcplength*
10104895	4971517

Tabel 3.8 Centroid hasil proses clustering

Centroid	*totlength*	*tottcplength*
1	11922.45455	4364.136364
2	983613	355522
3	10104895	4971517

Setelah didapatkan masing-masing nilai *centroid*, langkah selanjutnya adalah menentukan nilai *SSW* (*Sum of Square Within Cluster*) untuk mengevaluasi densitas data dengan *centroid*-nya. Perhitungan *SSW* didapatkan dengan menghitung jarak setiap data pada *centroid* dan dihitung rata-ratanya. Hasil perhitungan lengkap SSW ditunjukan pada **tabel 3.9.**

Tabel 3.9. SSW

SSW_1	122792.3991
SSW_2	0
SSW_3	0

Setelah nilai *SSW* didapatkan, langkah berikutnya adalah melakukan perhitungan untuk nilai *SSB* (*Sum of Square Between Cluster*) sebagai ukuran keterpisahaan antar *cluster,* untuk mendapatkan nilai *SSB* dilakukan dengan menghitung jarak antar *centroid* suatu *cluster*. Tabel *SSB* ditunjukan oleh **tabel 3.10**.

Tabel 3.10 SSB

$SSB_{1,2}$	1033196.187
$SSB_{1,3}$	11249031.17
$SSB_{2,3}$	10222778.25

Setelah nilai *SSW* (densitas) & *SSB* (separasi) didapatkan, langkah selanjutnya adalah melakukan evaluasi rasio (R_{ij}) yang bertujuan mendapatkan nilai *DBI* untuk tiap *cluster*. *Cluster* yang baik adalah *cluster* yang memiliki nilai densitas sekecil & nilai separasi sebesar mungkin, nilai rasio (*DBI*) yang dimiliki oleh masing-masing *cluster* tersebut digunakan untuk mengevaluasi *DBI* keseluruhan *cluster*. Perhitungan *R* & *DBI* ditunjukan pada **tabel 3.11**.

Tabel 3.11. R & DBI

	R	**Jam ke -*i***			***R Max***	***DBI***
		1	**2**	**3**		
Jam ke -*i*	1	0	0.118847128	0.01091582	0.118847	0.08287
	2	0.118847128	0	0	0.118847	
	3	0.01091582	0	0	0.010916	

Nilai R_{ij} yang didapatkan pada tiap *cluster* menunjukan nilai *index davies bouldin* yang menjelaskan rasio densitas data pada tiap *cluster* adalah 0.118, 0.010, 0. Rasio dengan nilai terbesar dipilih untuk dicari rata-ratanya sehingga menghasilkan nilai DBI 0.082.

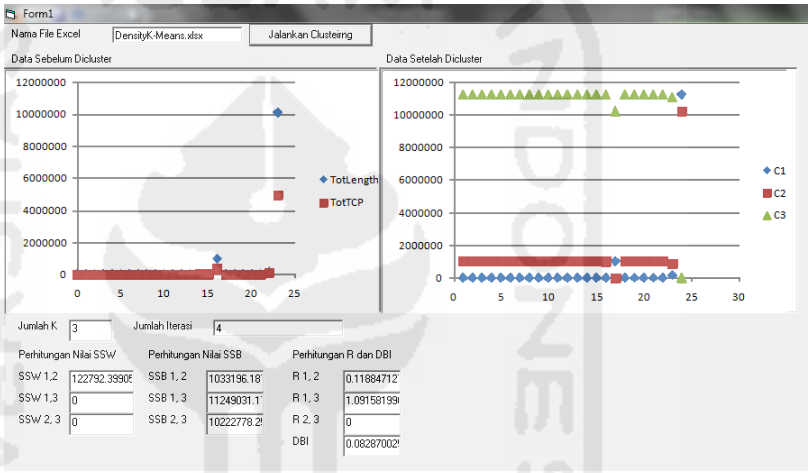
3.2.6. Penerapan Sistem

Pada bagian ini akan menjelaskan tahapan-tahapan proses dalam melakukan *clustering* menggunakan *Density K-Means*. Tahapan dalam

melakukan clustering dibagi menjadi 3 (tiga) proses, diantaranya adalah dijelaskan sebagai berikut:

a. Input

Dalam menerapkan perancangan sistem, aplikasi sederhana yang digunakan untuk melakukan proses *clustering* menggunakan VB6.0, langkah pertama adalah merancang antarmuka yang ditunjukan **gambar 3.15**.



Gambar 3.15. Tampilan Antarmuka

Proses diawali dengan membaca data dari berkas *excel* yang berisi frekuensi paket data atau fitur yang akan diproses menggunakan algoritma *density K-means*. Dalam berkas excel tersebut terdapat fitur *totlength* dan *tottcplength* dan perhitungan manual *Density k-means* dan validasi internal menggunakan *DBI*.

b. Proses

Pada bagian ini proses *clustering* mengacu pada proses *clustering* pada poin sebelumnya mengenai penerapan *clustering* yang dilakukan dengan perhitungan manual. Maka pembuatan aplikasi dilakukan dengan cara meng-*embedded* berkas excel yang berisi perhitungan manual agar dapat ditampilkan oleh masing-masing komponen dalam antarmuka yang telah disediakan. Proses dimulai

dengan menentukan jumlah *K* dan iterasi yang akan dibutuhkan dalam proses *clustering*, dalam hal ini jumlah *K* dan iterasi sudah ditentukan masing-masingnya 3 (tiga) dan 4 (empat). Setelah kedua parameter ini diberikan, lalu tekan tombol *clustering* menjalankan proses *clustering*.

c. Output

Ketika aplikasi telah selesai melakukan pemrosesan, aplikasi akan menghasilkan *Output* hasil perhitungan manual untuk tiap-tiap iterasi, dan proses melakukan validasi internal pada data yang menjadi anggota atau mengikuti suatu *cluster*. hasil dari aplikasi yang menggunakan teknik *embedded*, akan menampilkan nilai *SSW* (*density*) dan *SSB* (separasi), *R* dan terakhir nilai *DBI*. serta grafik data setelah dilakukan *clustering*.