BAB III

METODE PENELITIAN

3.1. Studi Pustaka

Metode ini dilakukan dengan mempelajari konsep, teori dan materi dari buku serta literatur lainnya yang mengarah kepada pemecahan masalah.

Pada tahap ini kita melakukan pembelajaran tentang sistem dari *firewall*, teori *fuzzy logic* dan *neural network* , dengan mengumpulkan literatur yang bisa kita dapatkan melalui *internet*, buku dan majalah yang secara khusus membahas tentang teori-teori yang disebutkan diatas.

3.2. Perancangan

Pada tahap ini dilakukan suatu perancangan sistem yang didasarkan atas teori-teori, informasi serta data-data yang telah terkumpul, sehingga setelah semua informasi telah terkumpul maka perancangan suatu sistem dapat segera dibuat.

Dalam hal ini pengumpulan informasi tentang perancangan sistem akan mengacu kepada hal-hal sebagai berikut :

1. Mengidentifikasi masalah-masalah kebutuhan *user*, contoh : *hardware* dan *software* apa saja yang akan digunakan untuk menunjang jalannya sistem secara sempurna.
2. Menyatakan secara spesifik sasaran yang ingin dicapai dalam perancangan suatu sistem yang diinginkan user, misalnya : dengan diterapkannya metode fuzzy logic dan neural network, sistem firewall diharapkan dapat melakukan pendeteksian serangan serta mengkategorikannya ke dalam berbagai jenis

tingkat serangan dan sistem juga dapat melakukan respon balik dengan sesuai dengan jenis serangan yang terjadi.

3. Memilih konfigurasi dan alternatif pemecahan masalah yang paling tepat untuk mendapatkan hasil maksimal untuk perancangan sistem.

Setelah semua informasi diatas telah terkonsep dengan baik, maka akan dilakukan perancangan sistem yang terbagi dalam beberapa tahap sebagai berikut :

3.2.1. Perancangan Firewall dengan IPTABLES

Firewall yang akan digunakan pada sistem ini adalah *packet filtering firewall* dengan menggunakan *IPTABLES* yang berjalan di sistem operasi LINUX dengan versi kernel diatas 2.4 .

Sedangkan untuk mendeteksi paket-paket data yang lewat digunakan sebuah *intrusion detection system (ids)*, dimana dalam perancangan kali ini model ids yang diterapkan adalah *host-based ids*. Untuk itu digunakan sebuah *tool* yaitu SNORT versi 2.3.3 yang berfungsi sebagai *sensor* dan *analyzer*. Sensor dan analyzer merupakan sistem deteksi dini dari sistem keamanan yang dirancang, yang berfungsi sebagai *intrusion detector* dan memiliki kemampuan *packet logging* serta analisa *traffic* secara *real time*. Hasil dari analyzer kemudian dimasukkan ke dalam database sehingga dapat dengan mudah di proses serta menjadi masukan bagi sistem yang lain.

Dalam sistem ini firewall dan SNORT akan di-*install* dalam *personal computer (pc)* yang sama, SNORT akan berfungsi sebagai pendeteksi paket-paket data yang kemudian akan diproses terlebih dahulu dengan software fuzzy dan

neural network dan kemudian firewall akan mengambil keputusan untuk memblok *source ip-address* yang mengirimkan paket abnormal tersebut (dalam hal ini disesuaikan dengan level serangan yang telah diolah dengan software fuzzy logic dan neural network).

A. IPTABLES

IPTABLES adalah modul di linux yang memberikan pengukuran langsung terhadap kernel linux mulai versi 2.4 yang diperuntukkan bagi keamanan sistem jaringan komputer. Konfigurasi IPTABLES paling sederhana setidaknya menangani tiga kumpulan aturan yang disebut *chain*. Paket-paket yang diarahkan ke firewall dinamakan *chain INPUT*, sedangkan paket yang diteruskan melewati firewall dinamakan *FORWARD* dan paket yang menuju jaringan eksternal meninggalkan firewall dinamakan *OUTPUT*.

Paket-paket yang masuk kemudian diperiksa dan diberikan ke *chain INPUT*. Tergantung pada informasi yang terdapat di dalam *header* paket dan kebijakan dalam *ruleset* keputusan yang diambil untuk suatu paket dapat berupa :

1. *ACCEPT*, yaitu menerima paket dan diproses lebih lanjut oleh kernel.
2. *DROP*, yaitu menolak paket tanpa pemberitahuan sama sekali.
3. *REJECT*, yaitu mengembalikan paket ke asalnya dengan pesan kesalahan ICMP.
4. *LOG*, yaitu melakukan *log* (pencatatan) terhadap paket yang bersesuaian.
5. *RETURN*, yaitu untuk *chain user-defined* akan dikembalikan ke *chain* yang memanggil, sedangkan untuk *chain INPUT, OUTPUT* dan *FORWARD* akan dijalankan kebijakan *default*.

6. Mengirim ke *chain user-defined*.

Keputusan kebijakan di atas dibuat oleh pernyataan *jump* yang terdapat di dalam *ruleset*. Keputusan-keputusan yang serupa dibuat dalam *chain FORWARD* dan *OUTPUT*.

Rule dalam IPTABLES dapat dikenakan terhadap asal paket (-s), tujuan paket (-d), protokol (-p), dan *port*. Misalnya, untuk menolak semua paket yang datang dari pc dengan *ip-address* 192.168.100.1, dapat ditulis :

```
iptables -t filter -A INPUT -s 192.168.100.1 -j DROP
```

atau

```
iptables -A INPUT -s 192.168.100.1 -j DROP
```

Dalam sistem ini, ip-address yang telah diblok secara otomatis oleh firewall akan mengalami masa *blocking* dengan acuan waktu, yaitu blocking dengan satuan jam dan hari, dimana hal ini disesuaikan dengan tipe serangan yang terjadi (*low attack, medium attack,* dan *high attack*). Kemudian untuk penghapusan *rule iptables* akan dilakukan secara otomatis dengan membandingkan waktu blocking dengan *localtime* server, apabila bernilai sama maka sistem akan melakukan penghapusan *rule iptables* terhadap ip-address yang terkena blocking.

3.2.2 Perancangan Sistem Fuzzy

Dalam pembuatan sistem fuzzy, masukan yang digunakan berasal dari informasi yang masuk ke alert IDS dalam hal ini dari *database* SNORT. Adapun tahap-tahap yang dilakukan antara lain :

A. *Input Fuzzy*

Memilih masukan (*input*) dari *field-field* yang ada di database SNORT yang akan dimasukkan ke dalam perhitungan fuzzy. Database SNORT sebenarnya terdiri dari 16 tabel *default* dan tiga tabel tambahan (*optional*). Akan tetapi dalam perancangan ini hanya lima tabel saja yang digunakan yaitu tabel iphdr, tcphdr, icmphdr, udphdr, dan data. Sedangkan dari tiap-tiap tabel tersebut selain tabel icmphdr dan udphdr, hanya beberapa field saja yang digunakan yaitu ip_src, ip_dst, data_payload, tcp_sport, tcp_dport, tcp_flags, tcp_ack, dan tcp_window. Alasan secara garis besarnya mengapa hanya menggunakan beberapa field tersebut adalah, karena field-field tersebut sudah mampu atau mencukupi untuk dijadikan sebagai indikator untuk membedakan antara kondisi data normal dan abnormal. Dibawah ini akan dijelaskan lebih lanjut tentang masukan (*input*) yang digunakan dalam perancangan ini sebagai *input fuzzy*, yaitu :

- ip_src digunakan untuk mengetahui alamat asal dari ip-address yang mencoba masuk ke dalam sistem.
- ip_dst digunakan untuk mengetahui alamat tujuan yang ingin dimasuki oleh suatu ip-address.
- data_payload digunakan untuk mengetahui data-data yang masuk ke dalam sistem selama koneksi berlangsung. Dalam perancangan ini fungsi data_payload difokuskan untuk memfilter *string-string* tertentu yang terkandung dalan suatu data yang masuk ke dalam sistem.

  tcp_sport digunakan untuk mengetahui alamat asal port yang digunakan oleh suatu user yang masuk ke dalam sistem melalui protokol tcp.

- tcp_dport digunakan untuk mengetahui alamat tujuan port yang ingin dituju atau dimasuki oleh user melalui protokol tcp.
- tcp_flags digunakan untuk mengetahui *flags-flags* apa saja yang aktif selama koneksi tcp berlangsung.
- tcp_ack digunakan untuk mengetahui besar nilai *acknowledgement number* yang ditampung selama koneksi berlangsung melalui protokol tcp. Setiap file yang terkirim melalui protokol TCP akan dipecah menjadi beberapa datagram dan masing-masing datagram tersebut mempunyai nomor urut (*sequence number*) yang berguna agar datagram tersebut dapat tersusun pada urutan yang benar. Sedangkan acknowledgement number yang bertugas untuk menunggu jawaban apakah datagram yang dikirim sudah sampai atau belum. Dalam sistem ini acknowledgement number (ack) merupakan indikasi tentang ada atau tidaknya suatu data yang masuk ke sistem.
- tcp_window digunakan untuk mengetahui besar nilai *window* yang ditampung selama koneksi berlangsung melalui protokol tcp. Window berfungsi untuk mengontrol berapa banyak data yang bisa singgah dalam satu waktu. Jika Window sudah terisi, ia akan segera langsung mengirim data tersebut dan tidak akan menunggu data yg terlambat, karena akan menyebabkan hubungan menjadi lambat. Dalam sistem ini window merupakan indikasi tentang ada atau tidaknya suatu data yang masuk ke sistem.
- Label icmphdr digunakan untuk mengetahui ada atau tidaknya protokol icmp yang aktif selama koneksi berlangsung. Dalam perancangan ini masukan fuzzy lebih ditekankan kepada jumlah total protokol icmp yang aktif dalam suatu koneksi.

- tabel udphdr digunakan untuk mengetahui ada atau tidaknya protokol udp yang aktif selama koneksi berlangsung. Dalam perancangan ini masukan fuzzy lebih ditekankan kepada jumlah total protokol udp yang aktif dalam suatu koneksi.

B. Fungsi Keanggotaan

Fungsi keanggotaan adalah bentuk representasi data-data *fuzzy* dalam variabel linguistik. Adapun data-data yang diambil dari tiap field diatas dikelompokkan ke dalam variabel linguistik dan dimasukkan ke dalam fungsi keanggotaan himpunan fuzzy. Dalam perancangan ini digunakan fungsi keanggotaan segitiga dan singleton. Dibawah akan dijelaskan lebih lanjut tentang fungsi keanggotaan dari masing-masing parameter yang digunakan, yaitu :

B.1. User, Port, Payload dan Flags

Parameter user, port, payload dan flags merupakan nilai-nilai yang mampu mereprentasikan kondisi yang terjadi di dalam suatu koneksi (pertukaran data maupun informasi) yang melibatkan hubungan antara user dan server (dilihat dari sisi user yang masuk, port yang digunakan, data payload yang masuk, dan flags apa saja yang aktif selama koneksi berlangsung). Sehingga dengan memasukkan nilai-nilai tersebut ke dalam sebuah parameter dan kemudian memasukkannya ke sebuah fungsi keanggotaan, diharapkan akan menghasilkan nilai-nilai yang unik dan bisa menjadi hasil analisa awal terhadap kondisi paket-paket data yang masuk ke dalam server.

Pemberian bobot untuk masing-masing parameter didasarkan atas kondisi-kondisi yang sering terjadi di dalam suatu jaringan komputer, dalam hal ini kebijakan yang dibuat menggambarkan kondisi yang terjadi di dalam sistem baik itu kondisi normal maupun kondisi abnormal yang melibatkan dua komputer saja, yaitu *client* dan server saja. Pemberian bobot untuk masing-masing parameter akan dijelaskan lebih lanjut dibawah ini :

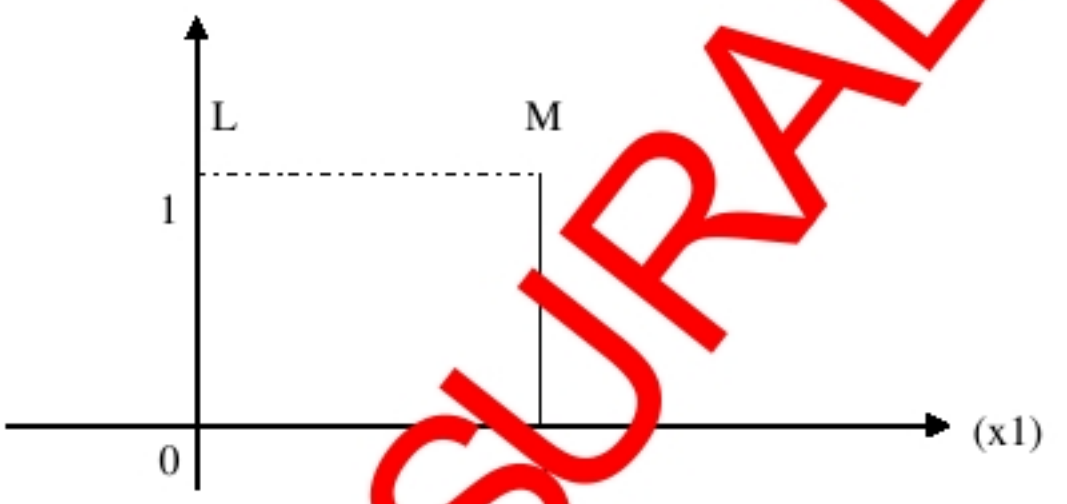
B.1.1. Fungsi Keanggotaan User

Fungsi keanggotaan dari parameter 'user' mewakili data dari field ip_src dan ip_dst yang terdapat pada tabel iphdr. Nilai fungsi keanggotaan user didapatkan dari pemberian bobot atas kondisi-kondisi yang berhubungan dengan kebijakan-kebijakan yang diterapkan untuk user yang masuk ke dalam sistem yang berfungsi sebagai server. Pemberian nilai bobot untuk user didasarkan atas kebijakan-kebijakan sebagai berikut :

- Bobot bernilai 0 yaitu bobot yang diberikan untuk menggambarkan kondisi user pada level *low* (rendah kemungkinannya terjadi sebuah serangan), nilai ini diberikan jika source ip-address yang masuk merupakan ip-address yang *trusted* (diperbolehkan mengakses server), dalam sistem ini ip-address yang diperbolehkan mengakses server adalah ip "192.168.100.1".
- Bobot bernilai 1 yaitu bobot yang diberikan untuk menggambarkan kondisi user pada level *medium* (indikasi terjadinya serangan akan tetapi pada tahap tidak begitu berbahaya), nilai ini diberikan jika source ip-address yang masuk merupakan ip-address yang *untrusted* (tidak diperbolehkan untuk mengakses

server), dalam sistem ini selain ip-address "192.168.100.1" maka ip-address yang lainnya dikategorikan sebagai *untrusted user*.

Sehingga didapatkan ambang batas seperti gambar 3.1 dibawah ini dimana nilai untuk kondisi normal akan masuk dalam kategori *low* dan kondisi abnormal di kategori *medium*. Adapun fungsi keanggotaan dari parameter ini adalah :



Gambar 3.1 Fungsi keanggotaan user

Jadi ada dua variabel linguistik untuk menyatakan parameter user (u), yaitu :

- Low (L)

  L(x1) = { 1, if u = = 0 }

  = { 0, if u < > 0 }

- Medium (M)

  M(x1) = { 1, if u = = 1 }

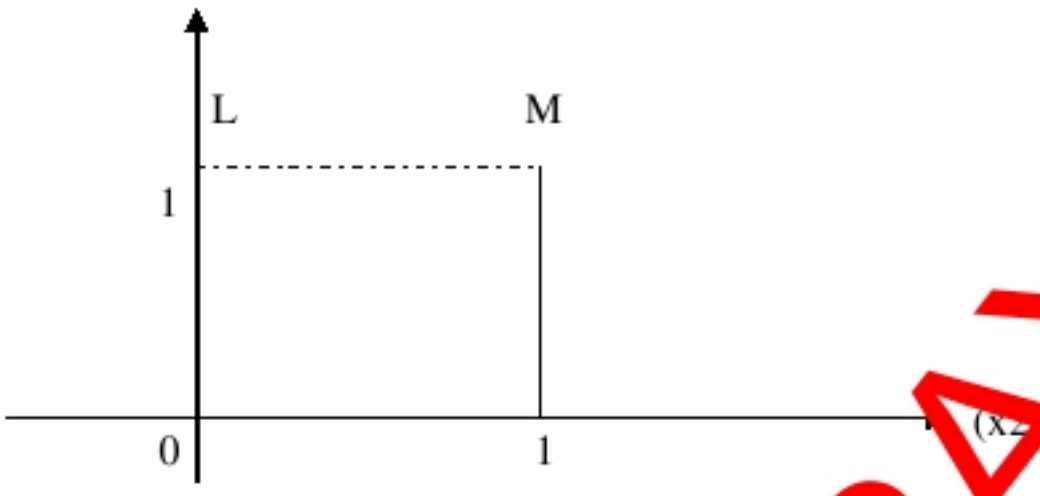
  = { 0, if u < > 1 }

B.1.2. Fungsi Keanggotaan Port

Fungsi keanggotaan dari parameter 'port' mewakili data dari field tcp_sport dan tcp_dport yang terdapat pada tabel tcphdr. Nilai fungsi keanggotaan port didapatkan dari pemberian bobot atas kondisi-kondisi yang berhubungan dengan kebijakan-kebijakan yang diterapkan untuk port-port mana saja di dalam server yang diperbolehkan untuk diakses atau dimasuki. Pemberian nilai bobot untuk port didasarkan atas kebijakan-kebijakan sebagai berikut :

- Bobot bernilai 0 yaitu bobot yang diberikan untuk menggambarkan kondisi port pada level *low* (rendah kemungkinannya terjadi sebuah serangan), nilai ini diberikan jika kondisi yang terjadi dilua semua kondisi dibawah ini.
- Bobot bernilai 1 yaitu bobot yang diberikan untuk menggambarkan kondisi port pada level *medium* (indikasi terjadinya serangan akan tetapi pada tahap tidak begitu berbahaya), nilai ini diberikan jika *destination port* yang dituju oleh user merupakan port 23 karena dalam sistem server ini penggunaan port 23 sebagai *service* untuk menjalankan telnet dilarang penggunaannya untuk user.

Sehingga didapatkan ambang batas seperti gambar 3.2 dibawah ini dimana nilai untuk kondisi normal akan masuk dalam kategori *low* dan kondisi abnormal di kategori *medium*. Adapun fungsi keanggotaan dari parameter ini adalah :



Gambar 3.2 Fungsi keanggotaan po

Jadi ada dua variabel linguistik untuk menyatakan para et r port (p), yaitu :

- Low (L)

  L(x2) = { 1, if p = = 0 }

  = { 0, if p < > 0 }

- Medium (M)

  M(x2) = { 1, if p = = 1 }

  = { 0, if p < > 1 }

B.1.3. Fungsi K anggot an Payload

Fungsi ke nggotaan dari parameter 'payload' mewakili data dari field data_payload yang terdapat pada tabel data. Nilai fungsi keanggotaan payload di apa pemberian bobot atas kondisi-kondisi yang berhubungan dengan kebijakan-kebijakan yang diterapkan untuk *filtering* data, dalam hal ini dilakukan penyaringan atas *string-string* yang terkandung dalam suatu informasi yang masuk ke dalam server. Pemberian nilai bobot untuk payload didasarkan atas kebijakan-kebijakan sebagai berikut :

- Bobot bernilai 0 yaitu bobot yang diberikan untuk menggambarkan kondisi payload pada level *low* (rendah kemungkinannya terjadi sebuah serangan), nilai ini diberikan jika kondisi yang terjadi diluar semua kondisi dibawah ini.
- Bobot bernilai 1 yaitu bobot yang diberikan untuk menggambarkan kondisi payload pada level *medium* (indikasi terjadinya serangan akan tetapi pada tahap tidak begitu berbahaya), nilai ini diberikan jika ditemukannya data payload yang mengandung string "port" maupun "open port". Umumnya bila terdapat string port maupun open port pada suatu data payload, berarti merupakan indikasi ada user yang mencoba melakukan *scanning port* untuk mengetahui port-port apa saja yang *open* (terbuka).
- Bobot bernilai 2 yaitu bobot yang diberikan untuk menggambarkan kondisi port pada level *high* (indikasi terjadi serangan pada tahap yang berbahaya), nilai ini diberikan jika ditemukannya data yang mengandung string "incorrect" maupun "login incorrect". Umumnya pesan kesalahan seperti incorrect maupun login incorrect berarti menunjukkan ada user yang mencoba-coba *login* ke dalam server akan tetapi gagal, umumnya dikarenakan kesalahan memasukkan *username* dan *password*.
- Bobot bernilai 3 yaitu bobot yang diberikan untuk menggambarkan kondisi payload pada level *very high* (indikasi serangan pada tahap yang sangat berbahaya), nilai ini diberikan jika ditemukan data payload yang mengandung string "port" maupun "open port" dan jika ditemukan data payload yang mengandung string "incorrect" maupun "login incorrect".

- Very High (VH)

  VH(x3) = { 1, if pa <= 3 }

  = { 0, if pa > 4 }

  = { (4-(pa (x3) / 1), if 3 < (pa (x3)) <= 4) }

B.1.4. Fungsi Keanggotaan Flags

Fungsi keanggotaan dari parameter 'flags' merupakan data dari field tcp_flags yang terdapat pada tabel tcphdr. Nilai fungsi keanggotaan flags didapatkan dari pemberian bobot atas kondisi-kondisi yang berhubungan dengan kebijakan-kebijakan yang diterapkan mengenai kondisi flags yang aktif selama koneksi berlangsung melalui protokol tcp. Pemberian nilai bobot untuk flags didasarkan atas kebijakan-kebijakan sebagai berikut :

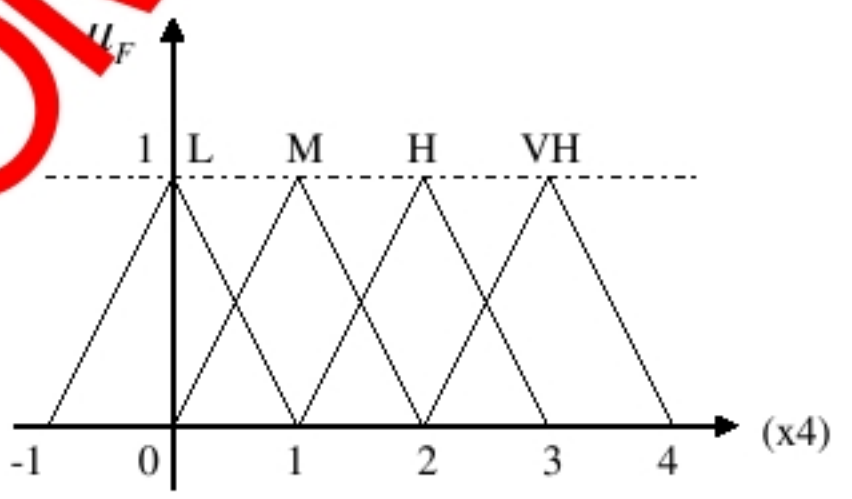
- Bobot bernilai 0 yaitu bobot yang diberikan untuk menggambarkan kondisi flags pada level *low* (kendali kemungkinannya terjadi sebuah serangan), nilai ini diberikan jika kondisi yang terjadi diluar semua kondisi dibawah ini.
- Bobot bernilai 1 yaitu bobot yang diberikan untuk menggambarkan kondisi flags pada level *medium* (indikasi terjadinya serangan akan tetapi pada tahap tidak begitu berbahaya), nilai ini diberikan jika flags dalam field tcp_flags bernilai 0 (tidak ada flags yang aktif), merupakan kondisi yang ilegal apabila satu paket data tidak mengaktifkan flags sama sekali (sering disebut sebagai *null packets*). (Bruneau, G . 2001 http://www.whitehats.ca)

  Bobot bernilai 2 yaitu bobot yang diberikan untuk menggambarkan kondisi flags pada level *high* (indikasi terjadi serangan pada tahap yang berbahaya), nilai ini diberikan jika flags yang aktif merupakan kombinasi flags FIN,

ataupun hanya flags FIN saja yang terus menerus aktif. Kombinasi flags SYN FIN adalah salah satu kombinasi yang ilegal, seperti kita ketahui flags SYN digunakan untuk memulai suatu hubungan atau koneksi sedangkan flags FIN untuk mengakhiri suatu koneksi. Sangat bersifat abnormal apabila kedua flags tersebut aktif secara bersamaan, begitu juga dengan kombinasi FIN dengan flags lainnya. (Bruneau, G . 2001 http://www.whitehats.ca)

- Bobot bernilai 3 yaitu bobot yang diberikan untuk menggambarkan kondisi flags pada level *very high* (indikasi serangan pada tahap yang sangat berbahaya), nilai ini diberikan jika tidak ada flags yang aktif padahal terjadi koneksi (pertukaran informasi / data) dan jika flags yang aktif merupakan kombinasi flags FIN, ataupun hanya flags FIN saja yang terus menerus aktif.

Sehingga didapatkan ambang batas seperti gambar 3.4 dibawah ini dimana nilai untuk kondisi normal akan masuk dalam kategori *low* dan kondisi abnormal di kategori *medium, high* dan *very high*. Adapun fungsi keanggotaan dari parameter ini adalah :



Gambar 3.4 Fungsi keanggotaan flags

Jadi ada empat variabel linguistik menyatakan parameter flags (f), yaitu :

- Low (L)

  L(x4) = { 1, if f <= 0 }

  = { 0, if f > 1 }

  = { (1-(f (x4) / 1), if 0 < (f (x4)) <= 1) }

- Medium (M)

  M(x4) = { 1, if f <= 1 }

  = { 0, if f > 2 }

  = { (2-(f (x4) / 1), if 1 < (f (x4)) <= 2) }

- High (H)

  H(x4) = { 1, if f <= 2 }

  = { 0, if f > 3 }

  = { (3-(f (x4) / 1), if 2 < (f (x4)) <= 3) }

- Very High (VH)

  VH(x4) = { 1, if f <= 3 }

  = { 0, if f > 4 }

  = { (4-(f (x4) / 1), if 3 < (f (x4)) <= 4) }

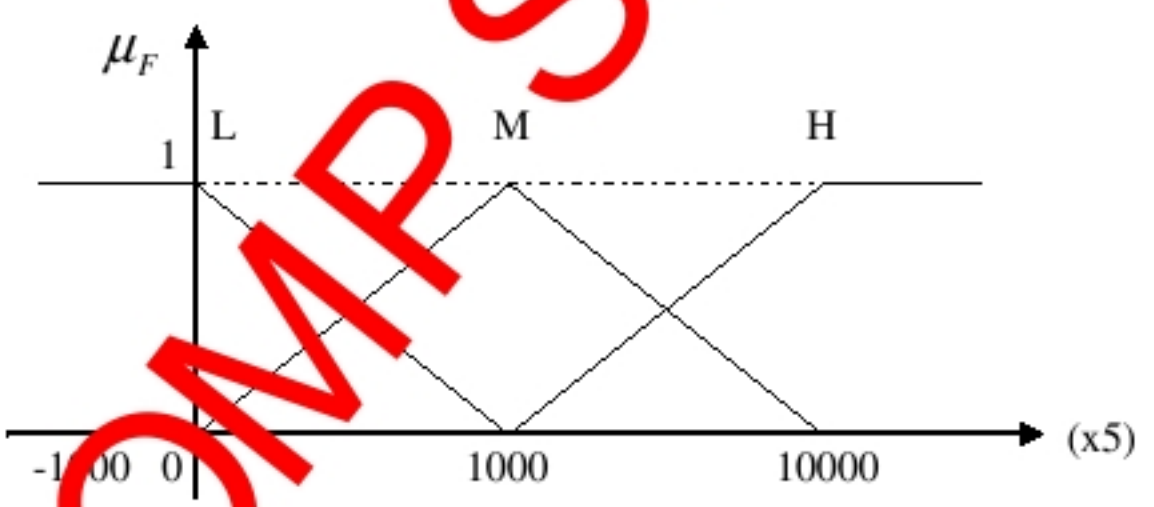
B.2. ACK dan Window

B.2.1. Fungsi Keanggotaan ACK

Fungsi keanggotaan dari parameter 'ACK' mewakili data dari field tcp_ack yang terdapat pada tabel tcphdr. Nilai fungsi keanggotaan ack mewakili nilai dari *acknowledgement number*. Dalam sistem firewall ini ack berfungsi sebagai indikasi tentang ada atau tidaknya paket yang masuk ke dalam server, dalam koneksi yang berjalan normal (melakukan *browsing data* dan *copy data*)

tercatat bahwa acknowledgement number memiliki nilai yang besar yaitu diatas $1x10^8$ (*unsigned int*), sedangkan ketika dilakukan serangkaian macam serangan nilai dari acknowledgement number memiliki nilai dari kisaran 0 sampai dengan dibawah $1x10^8$ (*unsigned int*). Dari pengamatan inilah, maka parameter ack digunakan. Untuk memudahkan perhitungan karena terdapat nilai ack yang terlampau besar (diatas 10^8), maka masukan nilai ack sebelum dimasukkan ke fungsi keanggotaan fuzzy, dilakukan normalisasi dengan cara nilainya dibagi dengan 10^5. Sehingga didapatkan ambang batas seperti gambar 3.5 dibawah ini dimana nilai untuk kondisi normal akan masuk dalam kategori *high* dan kondisi abnormal di kategori *low* dan *medium.*

Adapun fungsi keanggotaan dari parameter ini adalah :



Gambar 3.5 Fungsi keanggotaan ack

Jadi ada tiga variabel linguistik untuk menyatakan parameter ack (a), yaitu :

- Low (L)

L(x5) = { 1, if a <= 0 }

= { 0, if a > 1000 }

= { (1000-(a (x5) / 1000), if 0 < (a (x5)) <= 1000) }

- Medium (M)

  M(x5) = { 1, if a <= 1000 }

  = { 0, if a > 10000 }

  = {(10000-(a (x5) / 9000), if 1000 < (a (x5)) <= 10000) }

- High (H)

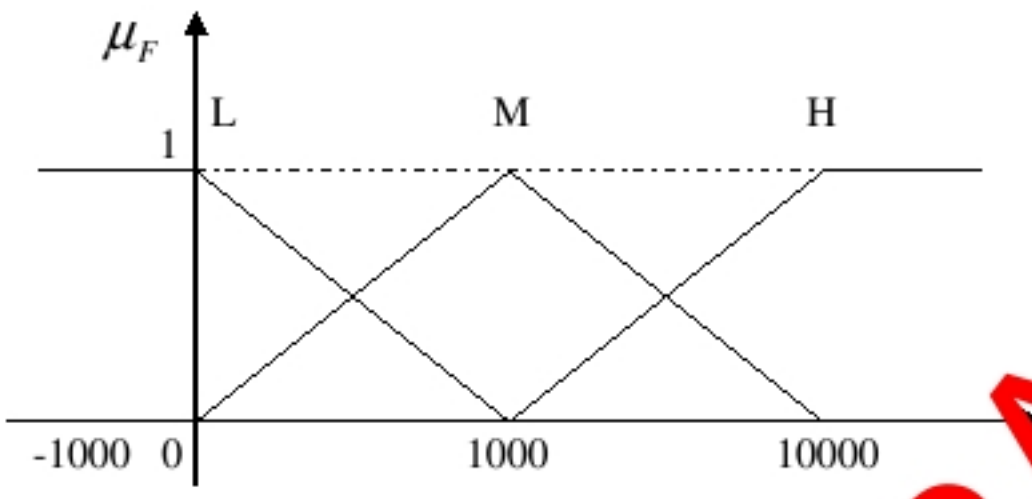
  H(x5) = { 1, if a >= 10000 }

  = { 0, if a > 100000 }

  = {(100000-(a (x5) / 90000), if 10000 < (a (x5)) <= 100000) }

B.2.2. Fungsi Keanggotaan Window

Fungsi keanggotaan dari parameter 'Window' mewakili data dari field tcp_window yang terdapat pada tabel tcp_hdr. Dalam sistem firewall ini window berfungsi sebagai indikasi tentang ada atau tidaknya paket yang masuk ke dalam server, dalam koneksi yang berjalan normal (melakukan *browsing data* dan *copy data*) tercatat bahwa window memiliki nilai diatas $1x10^5$ (*unsigned int*), sedangkan ketika dilakukan serangkaian macam serangan nilai dari window memiliki nilai dari kisaran $1x10^2$ sampai dengan dibawah $1x10^4$ (*unsigned int*). Dari pengamatan inilah, maka parameter window digunakan. Untuk memudahkan perhitungan agar bersesuaian nilainya dengan nilai ack maka dilakukan normalisasi dengan cara nilainya dibagi dengan 10. Sehingga didapatkan ambang batas seperti gambar 3.6 dibawah ini dimana nilai untuk kondisi normal akan masuk dalam kategori *high* dan kondisi abnormal di kategori *low* dan *medium*. Adapun fungsi keanggotaan dari parameter ini adalah :



Gambar 3.6 Fungsi keanggotaan wind

Jadi ada tiga variabel linguistik untuk menyatakan para et r window (w), yaitu :

- Low (L)

  L(x6) = { 1, if w <= 0 }

  = { 0, if w > 1000 }

  = { (1000-(w (x6) / 10), if 0 < (w (x6)) <= 1000) }

- Medium (M)

  M(x6) = { 1, if w <= 1 00 }

  = { 0, if w > 10000 }

  = (1000 (w (x6) / 9000), if 1000 < (w (x6)) <= 10000) }

- High (H)

  (x6) = { , if w >= 10000 }

  = { 0, if w > 100000 }

  = {(100000-(w (x6) / 90000), if 10000 < (w (x6)) <= 100000) }

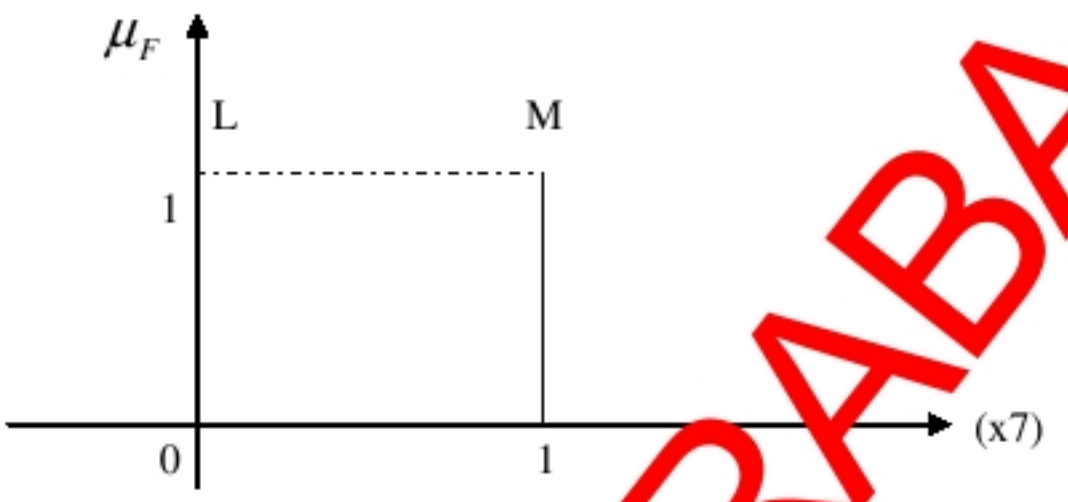
B.3. ICMP dan UDP

B.3.1. Fungsi Keanggotaan ICMP

Fungsi keanggotaan dari parameter 'ICMP' mengambil data dari tabel icmphdr. Nilai fungsi keanggotaan icmp didapatkan dari pemberian bobot atas kondisi-kondisi yang berhubungan dengan kebijakan-kebijakan yang diterapkan mengenai kondisi jumlah total protokol icmp yang aktif. Pemberian nilai bobot untuk icmp didasarkan atas kebijakan-kebijakan sebagai berikut:

- Bobot bernilai 0 yaitu bobot yang diberikan untuk menggambarkan kondisi icmp pada level *low*, nilai ini diberikan jika jumlah protokol icmp yang aktif dibawah 70. Nilai 70 didapatkan dari serangkaian hasil percobaan *icmp request* dengan cara melakukan ping ke komputer server dan mencoba serangan yang melibatkan protokol icmp. Dari hasil percobaan tersebut didapatkan kesimpulan bahwa dalam kondisi normal icmp request yang terjadi berada di kisaran 1 sampai dengan 10 icmp request, sedangkan untuk kondisi abnormal icmp request yang terjadi berjumlah diatas 70 icmp request.
- Bobot bernilai 1 yaitu bobot yang diberikan untuk menggambarkan kondisi icmp pada level *medium*, nilai ini diberikan jika jumlah protokol icmp yang aktif diatas 70. Jumlah protokol icmp yang melebihi 70 alert merupakan indikasi adanya seseorang yang mencoba membanjiri server dengan icmp request yang berlebihan sedangkan dalam kondisi normal misalnya ping, paling banyak hanya 10 icmp request yang terjadi.

Sehingga didapatkan ambang batas seperti gambar 3.7 dibawah ini dimana nilai untuk kondisi normal akan masuk dalam kategori *low* dan kondisi abnormal di kategori *medium*. Adapun fungsi keanggotaan dari parameter ini adalah :



Gambar 3.7 Fungsi keanggotaan icmp

Jadi ada dua variabel linguistik untuk menyatakan parameter icmp (i), yaitu :

- Low (L)

  L(x7) = { 1, if i = = 0 }

  = { 0, if i < > }

- Medium (M)

  M(x7) = { 1, if i = = 1 }

  = { 0, if i < > 1 }

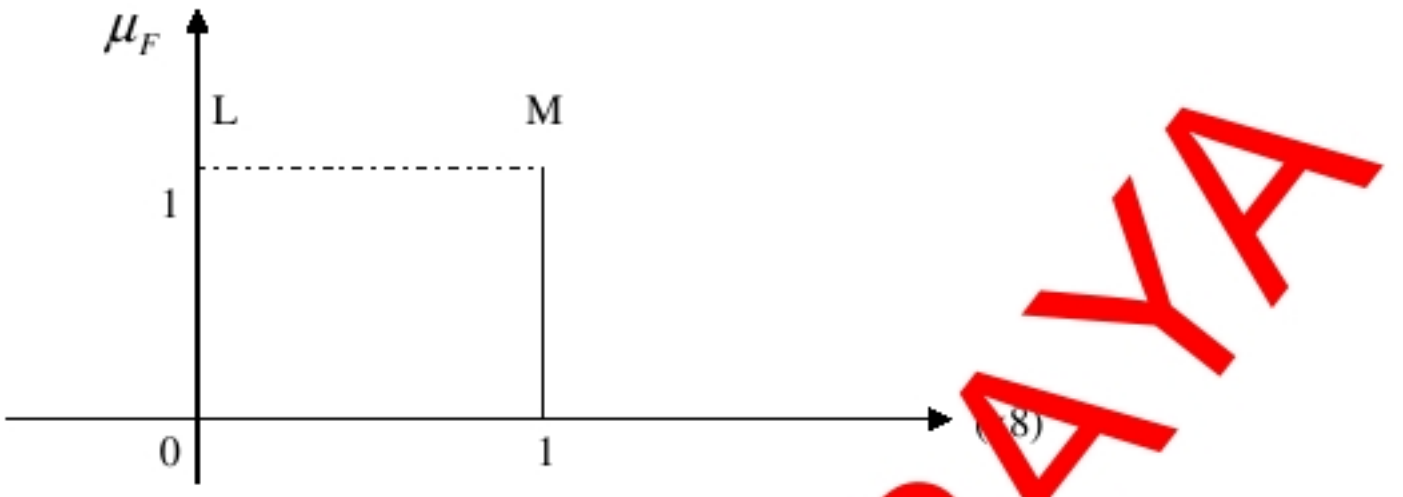
B.3.2. Fungsi Keanggotaan UDP

Fungsi keanggotaan dari parameter 'UDP' mengambil data dari tabel Udphdr. Nilai fungsi keanggotaan udp didapatkan dari pemberian bobot atas kondisi-kondisi yang berhubungan dengan kebijakan-kebijakan yang diterapkan

mengenai kondisi jumlah total protokol udp yang aktif. Pemberian nilai bobot untuk udp didasarkan atas kebijakan-kebijakan sebagai berikut :

- Bobot bernilai 0 yaitu bobot yang diberikan untuk menggambarkan kondisi udp pada level *low*, nilai ini diberikan jika jumlah protokol udp yang aktif dibawah 50. Nilai 50 didapatkan dari serangkaian hasil percobaan *udp request* dengan cara melakukan serangan yang melibatkan protokol udp. Dari hasil percobaan tersebut didapatkan kesimpulan bahwa dalam kondisi abnormal udp request yang terjadi berjumlah diatas 50 udp request, sedangkan untuk kondisi normal udp request yang terjadi di kisaran 1 hingga dengan 10 udp request.
- Bobot bernilai 1 yaitu bobot yang diberikan untuk menggambarkan kondisi udp pada level *medium*, nilai ini diberikan jika jumlah protokol udp yang aktif diatas 50. Jumlah protokol udp yang melebihi 50 alert merupakan indikasi adanya seseorang yang mencoba membanjiri server dengan udp request yang berlebihan.

Sehingga didapatkan ambang batas seperti gambar 3.8 dibawah ini dimana nilai untuk kondisi normal akan masuk dalam kategori *low* dan kondisi abnormal di kategori *medium*. Adapun fungsi keanggotaan dari parameter ini adalah :



Gambar 3.8 Fungsi keanggotaan udp

Jadi ada dua variabel linguistik untuk menyatakan parameter udp (ud), yaitu :

- Low (L)

  L(x8) = { 1, if ud = = 0 }

  = { 0, if ud < > 0 }

- Medium (M)

  M(x8) = { 1, if ud = = 1 }

  = { 0, if ud < > 1 }

C. Basis Aturan (*Rule Base*)

Setelah menentukan fungsi keanggotaan yang digunakan, maka sekarang perlu dibuat suatu kumpulan aturan-aturan yang menyatakan hubungan antar kondisi yang terjadi dan aksi apa yang harus dilakukan oleh *fuzzy* untuk menanggulangi segala kondisi yang terjadi. Basis aturan dinyatakan dalam *IF – THEN*, sebagai berikut :

C.1. Basis aturan untuk parameter user dan port

- If user = low and port = low Then usr_port = low
- If user = low and port = medium Then usr_port = medium
- If user = medium and port = low Then usr_port = medium
- If user = medium and port = medium Then usr_port = medium

Basis aturan untuk parameter diatas dapat diringkas dalam bentuk tabel basis aturan seperti tabel 3.1 dibawah ini :

Tabel 3.1 Basis aturan parameter 'usr' dan 'port'

usr \ port	Lpo	Mpo
Lu	L	M
Mu	M	M

Keterangan :

- Lu = Low user, yaitu parameter user dalam kondisi *low*.
- Mu = Medium user, yaitu parameter user dalam kondisi *medium*.
- Lpo = Low port, yaitu parameter port dalam kondisi *low*.
- Mpo = Medium port, yaitu parameter port dalam kondisi *medium*.

Parameter nilai user dan port untuk kondisi *low* dan *medium* menunjukkan seberapa besarnya bahaya yang terjadi di dalam sistem firewall bila dilihat dari kondisi user dan port. Kondisi *low* menyatakan aktivitas untuk parameter user dan port berlangsung dalam keadaan normal, sedangkan *medium*

menyatakan adanya kegiatan abnormal yang terjadi menyangkut ke dua parameter diatas.

C.2. Basis aturan untuk parameter payload dan flags

- If payload = low and flags = low Then payload_flags = low
- If payload = low and flags = medium Then payload_flags = medium
- If payload = low and flags = high Then payload_flags = medium
- If payload = low and flags = v_high Then payload_flags = very high
- If payload = medium and flags = low Then payload_flags = medium
- If payload = medium and flags = medium Then payload_flags = medium
- If payload = medium and flags = high Then payload_flags = high
- If payload = medium and flags = v_high Then payload_flags = very high
- If payload = high and flags = low Then payload_flags = medium
- If payload = high and flags = medium Then payload_flags = high
- If payload = high and flags = high Then payload_flags = high
- If payload = high and flags = v_high Then payload_flags = very high
- If payload = v_high and flags = low Then payload_flags = high
- If payload = v_high and flags = medium Then payload_flags = very high
- If payload = v_high and flags = high Then payload_flags = very high
- If payload = v_high and flags = v_high Then payload_flags = very high

Basis aturan untuk parameter diatas dapat diringkas dalam bentuk tabel basis aturan seperti tabel 3.2 dibawah ini :

Tabel 3.2 Basis aturan parameter 'payload' dan 'flags'

flags / payload	Lf	Mf	Hf	VHf
Lpa	L	M	M	VH
Mpa	M	M	H	VH
Hpa	M	H	H	VH
Vhpa	H	VH	VH	VH

Keterangan :

- Lpa = Low payload, yaitu parameter payload dalam kondisi *low*.
- Mpa = Medium payload, yaitu parameter payload dalam kondisi *medium*.
- Hpa = High payload, yaitu parameter payload dalam kondisi *high*.
- Vhpa = Very high payload, yaitu parameter payload dalam kondisi *very high*.
- Lf = Low flags, yaitu parameter flags dalam kondisi *low*.
- Mf = Medium flags, yaitu parameter flags dalam kondisi *medium*.
- Hf = High flags, yaitu parameter flags dalam kondisi *high*.
- Vhf = Very high flags, yaitu parameter flags dalam kondisi *very high*.

Parameter nilai payload dan flags untuk kondisi *low,medium, high* dan *very high* menunjukkan seberapa besarnya bahaya yang terjadi di dalam sistem firewall bila dilihat dari kondisi payload dan flags. Kondisi *low* menyatakan aktifitas untuk parameter payload dan flags berlangsung dalam keadaan normal, sedangkan *medium, high* dan *very high* menyatakan adanya kegiatan abnormal yang terjadi menyangkut ke dua parameter diatas.

C.3. Basis aturan untuk parameter ack dan window

- If ack = low and window = low Then ack_window = low
- If ack = low and window = medium Then ack_window = medium
- If ack = low and window = high Then ack_window = medium
- If ack = medium and window = low Then ack_window = medium
- If ack = medium and window = medium Then ack_window = medium
- If ack = medium and window = high Then ack_window = high
- If ack = high and window = low Then ack_window = medium
- If ack = high and window = medium Then ack_window = high
- If ack = high and window = high Then ack_window = high

Basis aturan untuk parameter diatas dapat diringkas dalam bentuk tabel basis aturan seperti tabel 3.3 dibawah ini :

Tabel 3.3 Basis aturan parameter 'ack' dan 'window'

ack \ window	Lw	Mw	Hw
La	L	M	M
Ma	M	M	H
Ha	M	H	H

Keterangan :

- La = Low ack, yaitu parameter ack dalam kondisi *low*.
- Ma = Medium ack, yaitu parameter ack dalam kondisi *medium*.
- Ha = High ack, yaitu parameter ack dalam kondisi *high*.

- Lw = Low window, yaitu parameter window dalam kondisi *low*.
- Mw = Medium window, yaitu parameter window dalam kondisi *medium*.
- Hw = High window, yaitu parameter window dalam kondisi *high*.

Berbeda dengan parameter-parameter sebelumnya, nilai ack dan window direpresentasikan hanya untuk tiga kondisi yaitu *low, medium* dan *high*. Ketiga-tiga kondisi tersebut menunjukkan seberapa besarnya bahaya yang terjadi di dalam sistem firewall bila dilihat dari kondisi ack dan window. Kondisi normal untuk ack dan window berada di level *high*, sedangkan *low* dan *medium* merupakan indikator kondisi abnormal.

C.4. Basis aturan untuk parameter 'icmp' dan 'udp'

- If icmp = low and udp = low Then icmp_udp = low
- If icmp = low and udp = medium Then icmp_udp = medium
- If icmp = medium and udp = low Then icmp_udp = medium
- If icmp = medium and udp = medium Then icmp_udp = medium

Basis aturan untuk parameter diatas dapat diringkas dalam bentuk tabel basis aturan seperti tabel 3.4 dibawah ini :

Tabel 3.4 Basis aturan parameter 'icmp' dan 'udp'

udp / icmp	Lud	Mud
Li	L	M
Mi	M	M

Keterangan :

- Li = Low icmp, yaitu parameter icmp dalam kondisi *low*.
- Mi = Medium icmp, yaitu parameter icmp dalam kondisi *medium*.
- Lud = Low udp, yaitu parameter udp dalam kondisi *low*.
- Mud = Medium udp, yaitu parameter udp dalam kondisi *medi*

Nilai icmp dan udp direpresentasikan hanya untuk 2 kondisi yaitu *low* dan *medium*. Ke-dua kondisi tersebut menunjukkan seberapa besarnya bahaya yang terjadi di dalam sistem firewall bila dilihat dari banyaknya aktivitas yang terjadi menggunakan protokol icmp dan udp. Kondisi normal untuk icmp dan udp berada di level *low*, sedangkan *medium* merupakan indikator kondisi abnormal.

D. Defuzzifikasi

Proses defuzzifikasi merupakan proses keluarnya nilai perhitungan antar parameter masukan atas kondisi yang terjadi di dalam suatu *traffic data* jaringan. Adapun hasil defuzzifikasi yang dicapai pada penelitian ini adalah dengan menggunakan metode *CA* (*Center Average*).

$$Y^{*} = \frac{\sum_{i=1}^{2} Y_i \, W_i}{\sum_{i=1}^{2} W_i}$$

Dimana Y* adalah keluaran *fuzzy* dalam bentuk *crisp* yang berupa nilai usr_port, payload_flag dan ack_window. Yi merupakan nilai tengah dari fungsi keanggotaan fuzzy, dan nilai Wi adalah tinggi dari derajat keanggotaan fuzzy.

Contoh : Menghitung nilai defuzzifikasi dari parameter ack dan window

- diperoleh masukan untuk parameter ack (x) = 1500
- diperoleh masukan untuk parameter window (x) = 2000
- digunakan rumus fungsi keanggotaan segitiga :

$$\mu_A(u) = \max\left[\min\left(\left(\frac{x-a}{b-a}\right),\left(\frac{c-x}{c-b}\right)\right),0\right] \qquad \ldots\ldots(4)$$

- dimana :

  x = masukan (*input*).

  a = batas awal kondisi fungsi keanggotaan

  b = batas kondisi fungsi keanggotaan yang bernilai 1.

  c = batas akhir kondisi fungsi keanggotaan

- sesuai dengan rumus diatas maka didapatkan perhitungan sebagai berikut :

$$low\ (\text{ack}) = \max\left[\min\left(\left(\frac{1500-(-1000)}{0-(-1000)}\right),\left(\frac{1000-1500}{1000-0}\right)\right),0\right]$$

$$medium\ (\text{ack}) = \max\left[\min\left(\left(\frac{1500-0}{1000-0}\right),\left(\frac{10000-1500}{10000-1000}\right)\right),0\right]$$

$$high\ (\text{ack}) = \max\left[\min\left(\left(\frac{1500-1000}{10000-1000}\right),\left(\frac{100000-1500}{100000-10000}\right)\right),0\right]$$

$$low\ (\text{ack}) = \max\left[\min\left(\left(\frac{2000-(-1000)}{0-(-1000)}\right),\left(\frac{1000-2000}{1000-0}\right)\right),0\right]$$

$$medium\ (\text{ack}) = \max\left[\min\left(\left(\frac{2000-0}{1000-0}\right),\left(\frac{10000-2000}{10000-1000}\right)\right),0\right]$$

$$high\ (\text{ack}) = \max\left[\min\left(\left(\frac{2000-1000}{10000-1000}\right),\left(\frac{100000-2000}{100000-10000}\right)\right),0\right]$$

- didapatkan hasil (pembulatan) :

  ack (*low*) = 0 window (*low*) = 0

  ack (*medium*) = 0.94 window (*medium*) = 0.88

  ack (*high*) = 0.05 window (*high*) = 0.11

- dihitung nilai tinggi hasil inferensi fuzzy dengan fungsi MIN

$$W_i = \min \left(x1_i , x2_i \right) \quad \ldots\ldots\ldots (2)$$

- defuzzifikasi dengan menggunakan CA :

$$Y^* = \frac{\sum_{i=1}^{2} Y_i W_i}{\sum_{i=1}^{2} W_i} \quad \ldots\ldots\ldots (3)$$

$Y i$ = nilai tengah dari masing-masing fungsi keanggotaan fuzzy.

$Y\,(low) = 500$

$Y\,(medium) = 5500$

$Y\,(high) = 55000$

- diperoleh hasil akhir / keluaran :

$Y^* = 15036.69724$

- hasil akhir dari defuzzifikasi ini agar masuk dalam hitungan interval 0 ~ 10 (dilakukan normalisasi agar sama intervalnya keluarannya dengan hasil defuzzifikasi parameter lainnya) maka perlu dibagi dengan 10000 dan dibulatkan, sehingga diperoleh hasil akhir :

  Pembulatan : $Y^* = \text{round}\ (1.5)$

  $Y^* = 1.$

3.2.3. Perancangan Neural Network

Metode pembelajaran neural network yang digunakan pada sistem ini adalah dengan metode *backpropagation*. Algoritma backpropagation biasanya diterapkan dalam *multilayer neural network*, akan tetapi kali penerapannya akan digunakan pada *single layer neural network*. Adapun pertimbangan pemilihan metode ini karena dapat diaplikasikan dalam berbagai bidang untuk melakukan pengenalan pola (*pattern recognation*), klasifikasi gambar, pengenalan suara dan telah banyak kesuksesannya dalam penerapannya di bidang teknologi komputer juga di bidang lainnya. Jaringan backpropagation mengadakan proses pembelajaran dengan pengaturan bobot hingga keluaran (*output*) yang dihasilkan jaringan sesuai dengan yang diharapkan dan di dalam sistem ini model *single layer* dengan algoritma backpropagation dirasakan sudah mampu untuk membedakan antara pola data normal dan abnormal. Karena menggunakan model single layer, maka lapisan yang digunakan hanya dua yaitu lapisan masukan (*input*) dan lapisan keluaran (*output*).

Agar jaringan dapat mengenali suatu pola tertentu, maka pola masukan harus dilatih (*training*) terlebih dahulu. Dan untuk mengetahui karakteristik jaringan maka perlu dilakukan pelatihan yang berulang agar dicapai suatu struktur jaringan yang tepat dalam proses pengelompokan keluaran. Pada proses pelatihan ini digunakan parameter seperti konstanta kecepatan belajar dan harga inisialisasi awal bobot yang bervariasi. Pada riset ini istilah belajar dan pelatihan pada single layer neural network adalah suatu proses pengaturan bobot-bobot sambungan antar simpul sehingga keluaran yang dihasilkan jaringan mendekati atau sama dengan keluaran yang diharapkan. Setelah mendapatkannya, maka akan

memperoleh konfigurasi jaringan yang diharapkan sehingga dapat melakukan proses pengujian. Pengujian dikatakan berhasil apabila jaringan sanggup menerima masukan dan menghasilkan keluaran seperti yang diharapkan. Adapun konfigurasi jaringan dalam tugas akhir ini, sebagai berikut:

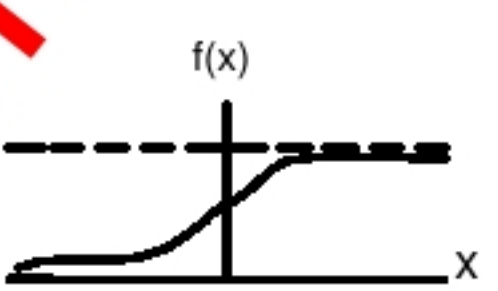
- Fungsi aktivasi

  Fungsi aktivasi jaringan yang dipilih adalah fungsi sigmoid biner. Fungsi tersebut dipilih dengan mempertimbangkan karakteristik yang dapat bernilai pada setiap titik dan bertambah secara monoton serta dapat berfungsi sebagai fungsi *squash* (dapat menerima masukan sampai tak terhingga nilainya dan menghasilkan keluaran pada kawasan terhingga).

  Fungsi sigmoid biner sebagai berikut :

$$f(x) = \frac{1}{1+\exp(-x)} \qquad \text{..... (1)}$$

Gambar fungsi sigmoid biner ditunjukkan pada gambar 3.9 dibawah ini :



Gambar 3.9 Fungsi sigmoid biner

- Proses pembelajaran

  Proses pembelajaran merupakan tahapan yang harus dilakukan agar neural network dapat mengenali pola-pola yang hendak diidentifikasikan. Beberapa

parameter proses yang mempunyai peran untuk menunjang keberhasilan tahap pembelajaran:

a. Konstanta belajar (α).

   Nilai konstanta belajar yang digunakan berada pada range antara 0 sampai dengan 1. Range nilai dipilih sesuai dengan beberapa percobaan yang telah dilakukan, yang dapat digunakan dalam pembuatan jaringan syaraf tiruan dengan metode pembelajaran backpropagation. Jika selisih dari pola pelatihan cukup besar maka dipilih nilai yang besar tetapi biasanya untuk pelatihan dengan pola yang kecil atau selisih antara pola satu dengan yang lainnya kecil maka dapat dipilih harga konstanta belajar yang kecil. Pada riset ini digunakan harga konstanta belajar optimal pada nilai 0.1.

b. Batas iterasi dan *error minimum*.

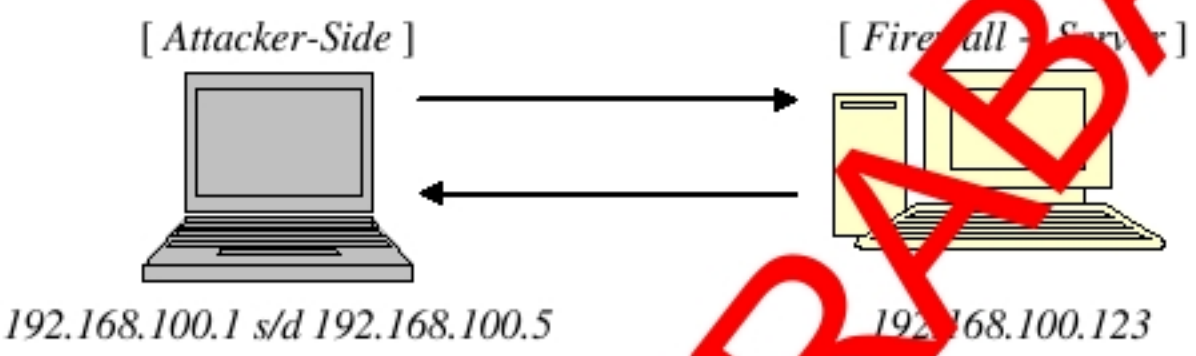
   Untuk menghindari proses pembelajaran terjebak dalam suatu *looping* yang terus-menerus maka diberikan batasan iterasi maksimum dan batasan error minimum. Pada riset ini digunakan batasan iterasi sebanyak 20000. Dan batasan error minimum 0.001.

c. Inisialisasi bobot.

   Bobot interkoneksi dari jaringan syaraf tiruan yang akan dilatih biasanya diinisialisaikan dengan harga yang kecil dan random. Inisialisasi mempunyai pengaruh yang besar terhadap hasil yang diinginkan. Nilai inisialisasi bobot berada pada range –0.5 sampai 0.5.

3.2.4. Perancangan Jaringan Komputer

Model jaringan komputer yang digunakan dalam penelitian ini terdiri dari 2 unit komputer yaitu 1 unit berfungsi sebagai *attacker* dan 1 unit sebagai firewall sekaligus sebagai server. Dua buah komputer tersebut terhubung melalui media kabel UTP (*Unshielded Twisted Pair*).



Gambar 3.10 Rancangan jaringan komputer

Alamat IP yang digunakan adalah IP kelas C yaitu 192.168.100.x dengan netmask 255.255.255.0 atau dengan jumlah *workstation* maksimal adalah 254 yaitu alamat 192.168.100.1 sampai dengan 192.168.100.254.

Untuk komputer server yang merangkap sebagai firewall digunakan IP 192.168.100.123 sedangkan untuk komputer *attacker* menggunakan IP 192.168.100.1 s/d 192.168.100.5 (untuk keperluan percobaan *attacking* maka komputer attacker dapat berganti IP Address). Untuk media transmisi digunakan kabel UTP dengan kecepatan transfer maksimum 100 mbps.

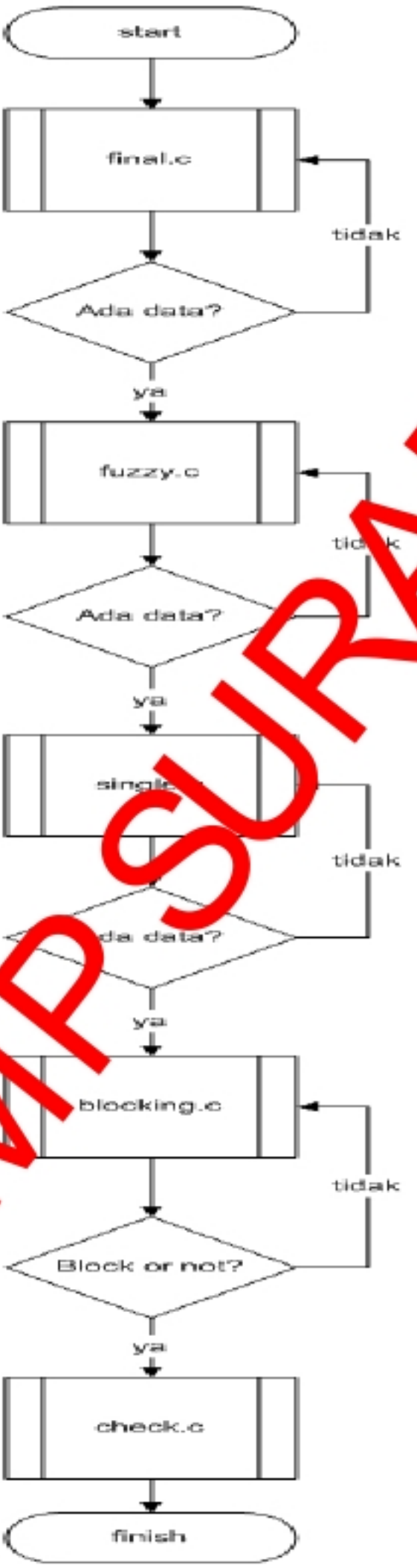
Pada tabel 3.5 dibawah ini akan dilampirkan spesifikasi dari ke-dua komputer yang digunakan :

Tabel 3.5 Spesifikasi hardware

Server + Firewall + IDS **(Personal Computer)**	**Attacker** **(Notebook ACER 4150NL[illegible]i)**
Processor AMD Thunder Bird 1 GHz	Processor Intel Pentium M 715 1[illegible]GHz
Memory DDR 384 (128 + 256) Mb	Memory DDR2 512 Mb
VGA 32 Mb (Share Memory)	VGA 128 Mb NVidia GeForce 6200
NIC Card XXX 10/100 (On Board)	NIC Card Broadcom 10/10[illegible]
Harddisk Maxtor 15 Gb	Harddisk Acer 1[illegible]60 GB
Operating System Linux Fedora Core 3	Operating System Linux Fedora Core 3

3.2.5. Perancangan Perangkat Lunak

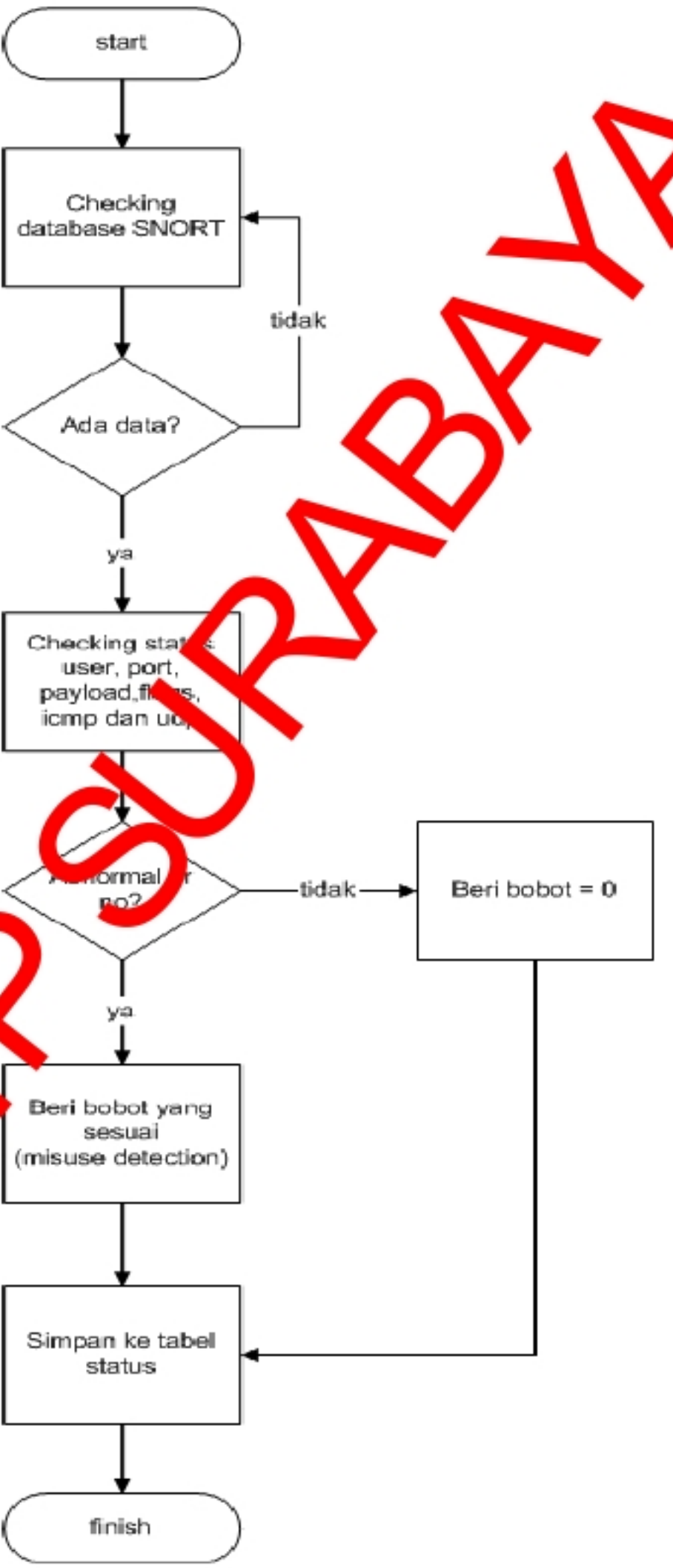
Dalam sistem ini, seluruh perangkat lunak (*software*) yang digunakan dan yang dikerjakan akan diterapkan (di *install*) pada server yang sekaligus berfungsi sebagai *packet filter firewall*. Sehingga seluruh proses *monitoring* dan pengendalian jaringan komputer akan dilakukan di server. Gambar 3.11 dibawah ini akan menggambarkan seluruh proses yang terjadi di dalam sistem :



Gambar 3.11 Flowchart sistem keseluruhan

A. final.c

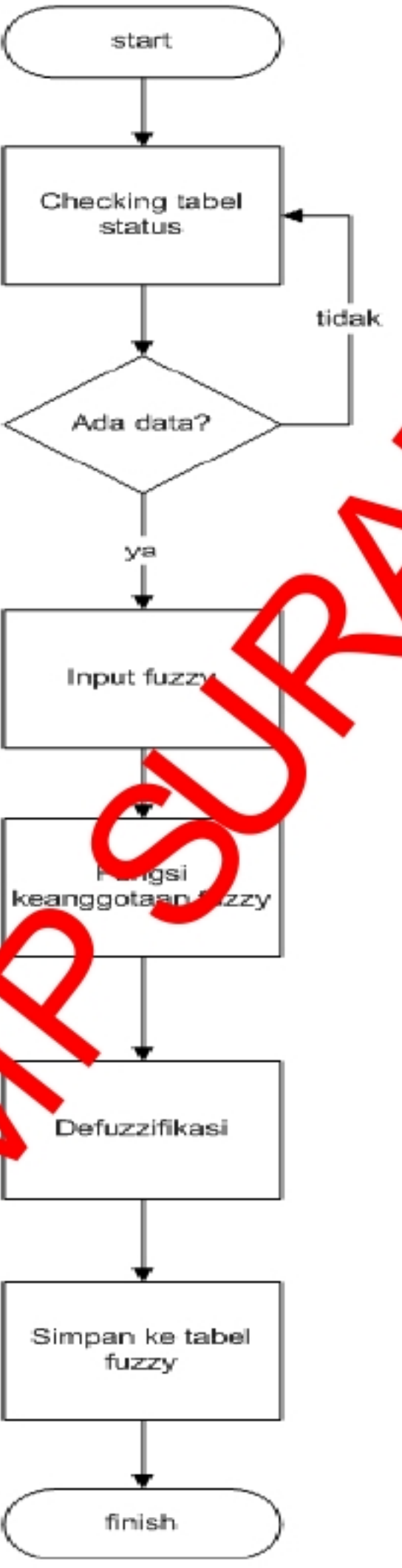
Program ini berfungsi untuk melakukan *query* ke database SNORT, yang bertujuan untuk mengambil nilai-nilai yang akan digunakan sebagai masukan (*input*) bagi software fuzzy. Di dalam program ini akan dilakukan pengecekan bagi kondisi-kondisi yang berkaitan dengan parameter-parameter seperti user yang mencoba masuk ke server, port-port apa saja yang digunakan untuk masuk ke server, string yang terkandung di dalam data payload, flags-flags apa saja yang aktif, serta jumlah protokol icmp dan protokol udp yang masuk ke dalam server. Selanjutnya akan dilakukan pemberian bobot untuk masing-masing kondisi yang terjadi. Untuk kondisi normal akan diberikan bobot = 0 (*null*), sedangkan untuk kondisi abnormal akan diberikan bobot dari kisaran 1 sampai dengan 3 (pemberian bobot kondisi abnormal disesuaikan dengan kebijakan-kebijakan yang diterapkan oleh sistem firewall). Keluaran (*output*) akhir dari software ini akan disimpan ke dalam tabel status. Gambar 3.12 dibawah ini akan menggambarkan seluruh proses yang terjadi di dalam software final.c :



Gambar 3.12 Flowchart software final.c

B. fuzzy.c

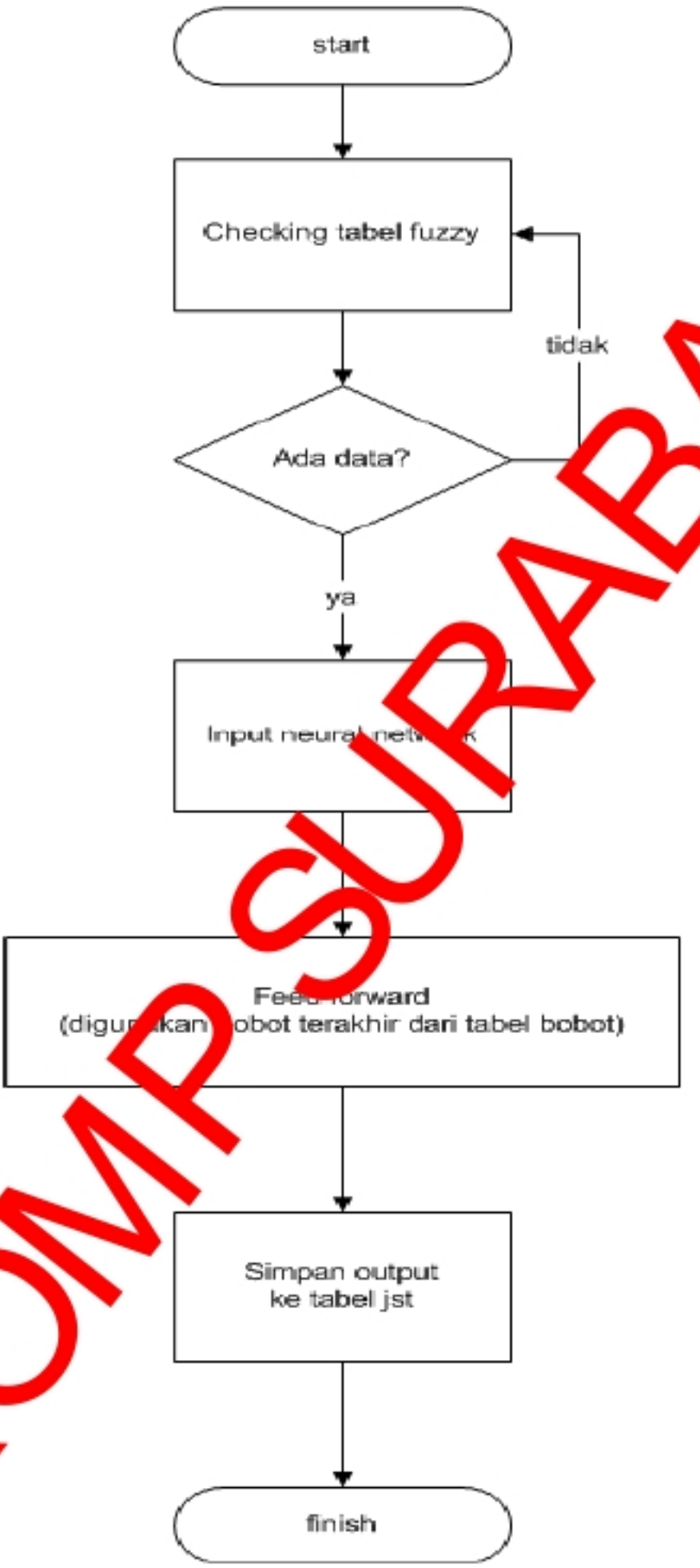
Program ini berfungsi untuk mengolah data-data hasil keluaran dari program final.c yang berada di tabel status dan kemudian memasukkannya ke dalam perhitungan dengan menggunakan metode fuzzy logic. Dimulai dari proses inisialisasi masukan (*input*) fuzzy, proses penerapan fungsi keanggotaan segitiga fuzzy, sampai dengan proses perhitungan defuzzifikasi dengan menggunakan metode *center average* (*CA*). Hasil keluaran dari perhitungan defuzzifikasi akan disimpan ke tabel fuzzy. Gambar 3.13 dibawah ini akan menggambarkan seluruh proses yang terjadi di dalam software fuzzy.c :



Gambar 3.13 Flowchart software fuzzy.c

C. single.c

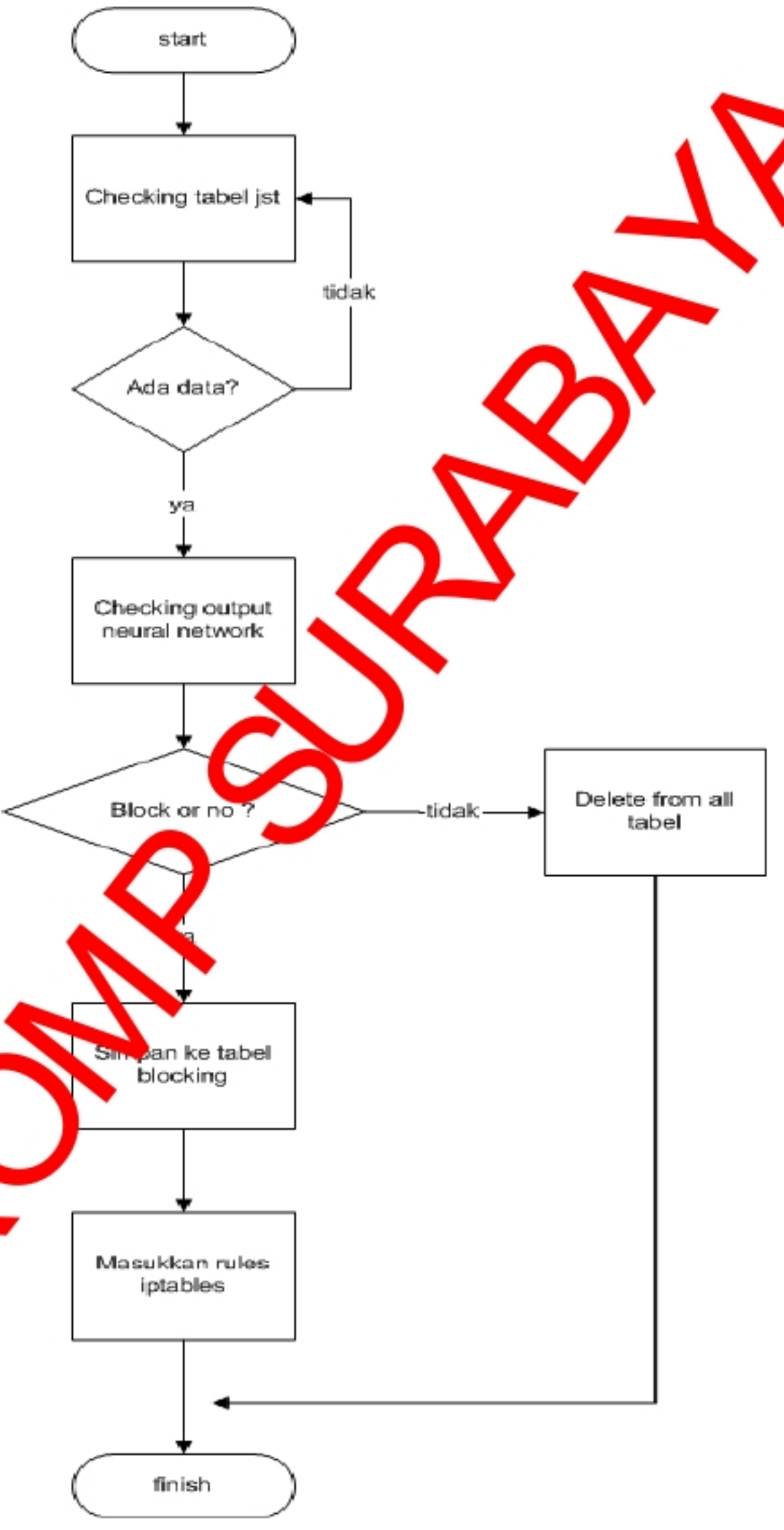
Program ini berfungsi untuk mengambil data hasil keluaran defuzzifikasi nilai fuzzy yang berada di tabel fuzzy untuk dijadikan sebagai masukan dalam proses *testing* menggunakan konsep *single layer neural network* dengan menggunakan algoritma *backpropagation*. Pada software ini hanya dilakukan perhitungan *feed forward* saja, karena software single.c berfungsi sebagai software untuk melakukan *testing phase* (tahap pengujian). Proses perhitungan feed forward dimulai dengan tahap pengambilan masukan dari tabel fuzzy lalu masuk ke tahap perhitungan dimana masing-masing masukan dikalikan dengan bobot terakhir dari proses pembelajaran, kemudian dikalikan dengan fungsi aktivasi *sigmoid biner* untuk menghitung keluaran. Hasil dari keluaran fungsi aktivasi yang merupakan keluaran jaringan (*output actual*) akan disimpan di tabel jst. Gambar 3.14 dibawah ini akan menggambarkan seluruh proses yang terjadi di dalam software single.c :



Gambar 3.14 Flowchart software single.c

D. blocking.c

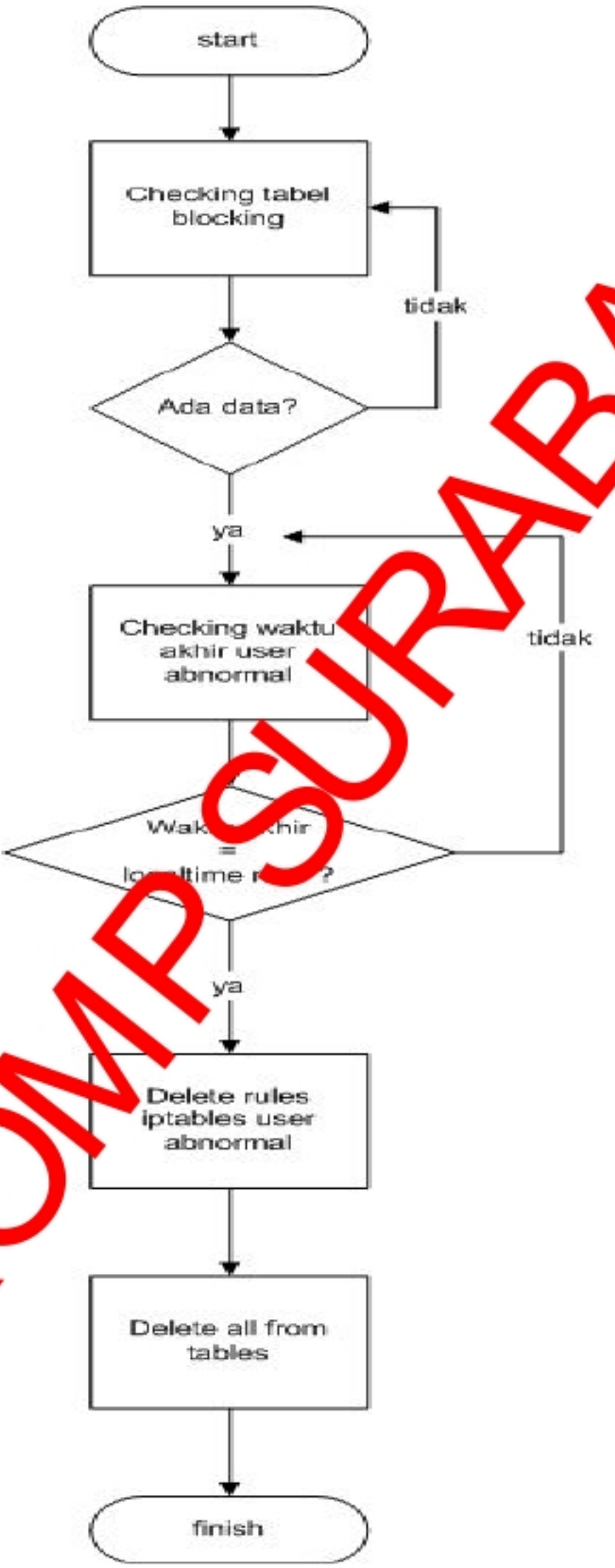
Program ini berfungsi untuk membaca data dari tabel jst yang merupakan keluaran dari metode feed forward, apabila ditemukan sebuah serangan (paket abnormal) maka akan dilakukan *blocking ip-address* yang disesuaikan dengan jenis serangan yang terjadi (*low attack, medium attack, high attack*) kemudian disimpan ke tabel blocking. Setelah itu akan dilakukan penambahan *rule iptables* sesuai dengan ip-address yang melakukan serangan. Dalam hal ini *rule* baru yang dimaksud adalah melakukan "*DROP source ip-address* " dengan satuan waktu. Misalnya melakukan *blocking ip-address* selama satu jam, satu hari dan tiga hari. Untuk kondisi normal maka tidak dilakukan blocking ip-address, hanya akan dilakukan penghapusan semua macam data-data yang berhubungan dengan ip-address yang berada pada kondisi normal tersebut. Gambar 3.15 dibawah ini akan menggambarkan seluruh proses yang terjadi di dalam software blocking.c :



Gambar 3.15 Flowchart software blocking.c

E. check.c

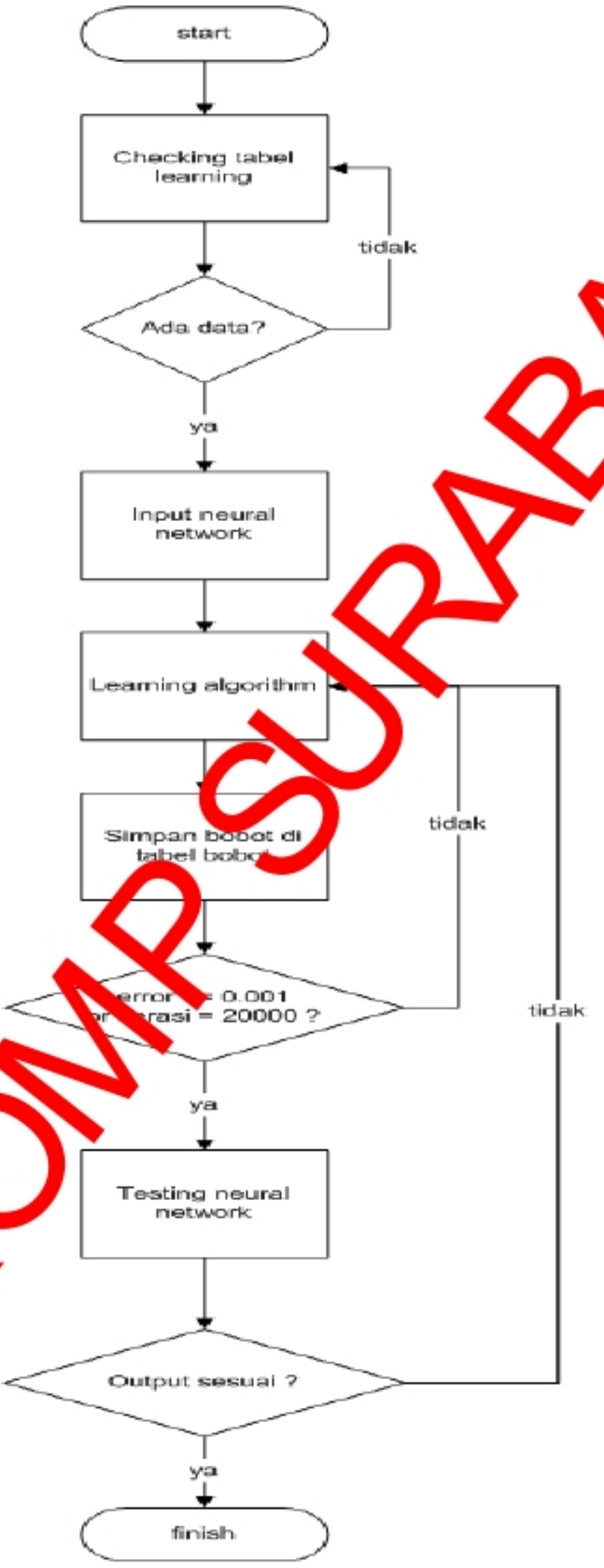
Program ini berfungsi untuk melakukan pengecekan terhadap ip-address yang statusnya masuk dalam kategori *blocking* yang berada di tabel blocking. Apabila waktu hukuman *blocking* sama dengan waktu dari *localtime* server maka akan dilakukan penghapusan *rule* IPTABLES yang sesuai dengan kondisi abnormal tersebut sekaligus penghapusan user yang berada di tabel blocking. Gambar 3.16 dibawah ini akan menggambarkan seluruh proses yang terjadi di dalam software check.c :



Gambar 3.16 Flowchart software check.c

F. learning.c

Program ini berfungsi untuk melakukan proses pembelajaran neural network dengan menggunakan metode backpropagation. Dalam tahap pembelajaran (*learning algorithm*) ini melibatkan beberapa tahap yaitu perhitungan feed forward, perhitungan error dengan backpropagation dan penyesuaian bobot (*update weight*). Pasangan pola masukan dan pola target disimpan di tabel learning, selama keluaran belum mencapai hasil yang diinginkan maka bobot yang digunakan untuk tahap pembelajaran ini diperbaharui (*update*) dan disimpan di dalam tabel bobot. Proses pembelajaran berhenti apabila nilai error dibawah atau sama dengan nilai *error minimum*, sedangkan untuk pembatasan agar proses pembelejaran tidak terjebak oleh *looping* yang lama, maka diberikan batasan jumlah iterasi sebanyak 20.000. Bobot terakhir hasil pembelajaran akan menjadi bobot untuk tahap *testing* (pengujian). Gambar 3.17 dibawah ini akan menggambarkan seluruh proses yang terjadi di dalam software learning.c :



Gambar 3.17 Flowchart software learning.c

3.3. Pembuatan

3.3.1. Instalasi dan Konfigurasi IDS (SNORT 2.3.3)

Langkah awal yang akan dilakukan untuk menginstall SNORT adalah terlebih dahulu kita sudah mendapatkan *packet installer* buat SNORT tersebut, dalam hal ini kita bisa men-*download* versi terbaru SNORT dari situs resmi SNORT di www.snort.org. Dalam sistem kali ini dipergunakan SNORT versi 2.3.3. Adapun langkah-langkah selanjutnya adalah sebagai berikut :

A. Instalasi SNORT

Setelah memperoleh paket SNORT-2.3.3.tar.gz , maka pertama kali yang dilakukan adalah :

1. Menempatkannya di sebuah direktori kemudian buka dengan perintah sebagai berikut :

```
# tar -zxvf snort-2.3.3.tar.gz
```

2. Setelah file SNORT-2.3.3.tar.gz terbuka, maka akan terbentuk folder baru bernama SNORT-2.3.3, kemudian langkah selanjutnya adalah mengakses folder tersebut dengan perintah :

```
# cd/snort-2.3.3
```

3. Selanjutnya lakukan perintah konfigurasi SNORT-* dengan perintah sebagai berikut :

```
#./configure--with-mysql
```

Untuk perintah ./configure yang ditambahkan dengan --with-mysql berarti SNORT akan dikonfigurasikan dengan database mysql. Adapun fungsi database mysql disini adalah untuk menyimpan alert SNORT.

4. Selanjutnya *compile source code* SNORT dengan menggunakan perintah ini :

```
# make
```

5. Setelah selesai meng-*compile*, maka tinggal menginstall software SNORT dengan perintah :

```
# make install
```

Secara default, SNORT akan terinstall ke dalam direktori /usr/local/bin.

B. Konfigurasi SNORT

Setelah melakukan proses penginstallan dengan *rules*, maka selanjutnya akan dilakukan beberapa konfigurasi terhadap SNORT. Adapun hal-hal yang perlu dikonfigurasi adalah sebagai berikut :

1. Download dan install *rules* terbaru di www.snort.org. Pastikan file *rules* yang didownload merupakan *snortrules.tar.gz* bukan yang *snortrules-current.tar.gz*. Setelah itu buat folder baru di /etc dengan nama snort, langkah selanjutnya *copy* file *snortrules.tar.gz* tersebut ke folder /etc/snort dan buka dengan perintah *tar*, hal diatas dapat dijabarkan dengan perintah :

```
# mkdir /etc/snort
# cp snortrules.tar.gz /etc/snort
# cd /etc/snort
# tar -zxvf snortrules.tar.gz
# cd /etc/snort/rules
# mv * ../
# cd ..
# rmdir rules
```

Isi dari *rules* yang terdapat di direktori /etc/snort/rules dipindahkan semua ke direktori /etc/snort supaya tidak perlu lagi mengkonfigurasi variabel *$RULE_PATH* yang terdapat di file snort.conf.

2. Konfigurasi file snort.conf yang terdapat pada direktori /etc/snort/snort.conf

- Konfigurasi keluaran (*output*) database yang diinginkan, *user*, password pengguna SNORT dan host dimana SNORT berjalan adalah sebagai berikut :

```
# database: log to a variety of databases
# See the README.database file for more information about configuring
# and using this plugin.
output database:log,mysql, user=root password=test dbname=db  localhost
                          Dikonfigurasi menjadi : →
output database:log,mysql, user=root password=munty123  dbn  snort
host=localhost
# output database: alert, postgresql, user=snort dbname=sno
# output database: log, odbc, user=snort dbname=sn rt
# output database: log, mssql, dbname=snort us =sno  password=test
# output database: log, oracle, dbname= ort u  t password=test
```

- Konfigurasi variabel *$RULE_PATH* yaitu dengan menghapus semua variabel tersebut, sedangkan untuk konfigurasi lainnya adalah mengaktifkan dan menonaktifkan *rules* sehingga didapatkan hasil sebagai berikut :

```
# Include all relevant rulesets here
#
# The following rules are disabled by default:
#
# web-attacks, backdoor, shellcode, policy, porn, info, icmp-info, virus,
# chat, multimedia, and p2p
#
# These rules are either site policy specific or require tuning in order to
n
# generate false positive alerts in most enviornments.
#
# Please read the specific include file for more information and
# README.alert_order for how rule ordering affects how alerts are triggered.
#--------------------------------------


include local.rules
```

```
#include bad-traffic.rules

#include exploit.rules

#include scan.rules

#include finger.rules

#include ftp.rules

#include telnet.rules

#include rpc.rules

#include rservices.rules

#include dos.rules

#include ddos.rules

#include dns.rules

#include tftp.rules


#include web-cgi.rules

#include web-coldfusion.rules

#include web-iis.rules

#include web-frontpage.rules

#include web-misc.rules

#include web-client.rules

#include web-php.rules


#include sql.rules

#include x11.rules

#include icmp.rules

#include netbios.rules

#include misc.rules

#include attack-responses.rules

#include oracle.rules

#include mysql.rules

#include snmp.rules


#include smtp.rules

#include imap.rules

#include pop2.rules

#include pop3.rules


#include nntp.rules
```

```
#include other-ids.rules
#include web-attacks.rules
#include backdoor.rules
#include shellcode.rules
#include policy.rules
#include porn.rules
#include info.rules
#include icmp-info.rules
#include virus.rules
#include chat.rules
#include multimedia.rules
#include p2p.rules
#include experimental.rules
```

3. Buat direktori untuk menyimpan hasil log dari SNORT, dengan perintah :

```
# mkdir /var/log/snort
```

4. Konfigurasi file /etc/init.d/rc.local atau rc.local agar snort dapat berjalan otomatis saat pertama kali komputer dinyalakan (*booting*), lakukan dengan perintah :

```
# /usr/bin/snort -U -i eth0 -D -c /etc/snort/snort.conf
```

5. Konfigurasi terakhir adalah melakukan *setting* terhadap database SNORT, dimana dalam sistem ini digunakan database MySQL. Secara *default* paket MySQL sudah terinstall saat pertama kali menginstall Linux, apabila belum terinstall maka dapat diperoleh di cd installer Linux atau dapat juga mendownloadnya di situs www.mysql.com. Biasanya paket MySQL bertipe rpm, jadi lebih mudah untuk diinstall dengan perintah sebagai berikut :

```
# rpm -ivh MySQL-*.X-X.i386.rpm
```

6. Setelah MySQL sudah terinstall, maka login ke MySQL sebagai root dan buat database baru bernama snort dan lakukan beberapa macam konfigurasi sebagai berikut ini :

```
# mysql -u root
```

```
# mysql> set password for 'root'@'localhost'=password('munty1234');
# mysql> create database snort;
# mysql> connect snort;
# mysql> source create_mysql;
# mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to snor
# mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* t
  snort@localhost;
# mysql> connect mysql;
# mysql> set password for 'snort'@'localhost'=password('m      1234
# mysql> set password for 'snort'@'%'=password('muntyl    4');
# mysql> flush privileges;
# mysql> exit;
```

Source create_mysql merupakan suatu peri h u k mengeksekusi *script create_mysql*, yaitu *script* untuk me buat tabel beserta *field-field* di dalamnya, lengkap dengan tipe data untu atab e SNORT. Sehingga dengan menggunakan *script create_my* an lebi memudahkan supaya kita tidak bersusah payah untuk membu abel, eld serta tipe data yang berhubungan dengan keluaran S ORT. Script ini bisa didownload di alamat http://cvs.sourceforge.n cgi-bin/viewcvs.cgi/snort/snort/contrib/ .

3.3.2. Konfigura IPTABLES

aket iptab es pada umumnya telah terinstall secara otomatis pada waktu melak kan instalasi sistem operasi, namum jika belum terinstall lakukan proses insta secara manual, dimana paket source iptables dapat diperoleh pada cd in ller sistem operasi linux yang digunakan.

Ketika suatu aturan sudah diberikan pada firewall, secara otomatis firewall akan menjalankan aturan tersebut untuk diterapkan pada paket yang datang, melintas dan keluar dari firewall. Hal ini disebabkan, perintah iptables

langsung memodifikasi *chain* aturan yang ada pada kernel. Beberapa konfigurasi yang dilakukan pada iptables adalah sebagai berikut :

A. Membuat Inisialisasi

Inisialisasi aturan iptables digunakan untuk membuat kebijakan umum terhadap *chain* iptables yang diterapkan pada firewall. Kebijakan ini akan diterapkan jika tidak ada yang sesuai. Kebijakan umum yang diterapkan adalah sebagai berikut :

```
#  iptables -P INPUT ACCEPT
#  iptables -P FORWARD ACCEPT
#  iptables -P OUTPUT ACCEPT
```

kebijakan diatas akan untuk memperbolehkan semua paket yang masuk dan keluar ke firewall serta menerima paket yang meninggalkan firewall.

3.3.3. Perangkat Lunak

A. final.c

Program ini berfungsi untuk mengambil nilai-nilai yang berada di dalam database snort. Khususnya data yang berada di dalam tabel iphdr, tcphdr, data untuk mengetahui ip address user yang masuk beserta paket-paket data yang ikut masuk bersamanya. Program ini kemudian memasukkan nilai-nilai tersebut ke dalam tabel status. Di dalam program ini terdapat pula pemberian bobot atas data-data yang masuk ke database snort.

Dibawah akan dijelaskan tabel dan field-field yang digunakan oleh sistem :

A.1. Tabel iphdr

Tabel iphdr berisi informasi mengenai atribut yang berhubungan dengan ip-address. Sedangkan di dalam program ini tidak semua field di dalam tabel iphdr digunakan, akan tetapi cuma field ip_src dan ip_dst, yaitu field yang berisi informasi mengenai alamat dari ip-address yang masuk dan tujuan yang dituju oleh ip-address tersebut. Fungsi program final.c disini adalah memberikan nilai bobot atas beberapa kejanggalan yang terjadi pada nilai-nilai field tersebut, kemudian memasukkannya ke tabel "status" dalam field "usr_status", yang berarti memperlihatkan status user yang coba mengakses server. Nilai bobot yang diberikan antara lain :

- usr bernilai 0 jika ip_src merupakan ip '192.168.100.1'.
- usr bernilai 1 jika ip_src bukan merupakan ip '192.168.100.1'.

Penggalan *listing code* untuk kondisi diatas adalah sebagai berikut :

```
// koneksi ke database MySql menggunakan C //
connection=mysql_real_connect(&mysql,"localhost","root","munty1234","snort",0,0,0)
;
// mengambil data dari field ip_src = ip_masuk dan ip_dst = 192.168.100.123 //
state=mysql_query(connection, "select c.*,inet_ntoa(b.ip_src),inet_ntoa(b.ip_dst)
from iphdr b inner join iphdr c on b.sid = c.sid and b.cid = c.cid where
inet_ntoa(ip_dst)='192.168.100.123'");
res=mysql_store_result(connection);
```

Kemudian selama field tidak kosong maka akan dilakukan pengecekan nilai dari tiap field yang dimaksud untuk melihat ada atau tidaknya suatu kondisi yang diinginkan.

```
while ((row=mysql_fetch_row(res))!=NULL)
{
    char ip[15];        strcpy(ip,row[12]);
    char me[13];        strcpy(me,"192.168.100.1");
    int compare_me;     compare_me=strcmp(ip,me);
    if(compare_me==0)  // bobot bernilai 0 bila ip_src = 192.168.100.1
    {
        // memasukkan nilai bobot = 0 untuk field usr //
        strcpy(query,"insert into status (
        usr_status,port_status,payload_status,flag_status,tcp_ack,tcp_window,ip_src
        ) values ('0','0','0','0','0','0','");
        strcat(query,ip); strcat(query,"')");
        state=mysql_query(connection,query);
    }
```

```
    else    // bobot bernilai 0 bila ip_src != 192.168.100.1
    {
        // memasukkan nilai bobot = 1 untuk field usr //
        strcpy(query,"insert into status (
        usr_status,port_status,payload_status,flag_status,tcp_ack,tcp_window,ip_src
        ) values ('1','0','0','0','0','0','");
        strcat(query,ip); strcat(query,"')");
        state=mysql_query(connection,query);
    }
}
```

A.2. Tabel tcphdr

Tabel tcphdr berisi informasi mengenai atribut yang berhubungan dengan segala macam koneksi yang melalui protokol TCP. Dari tabel ini akan diambil nilai-nilai dari field tcp_sport, tcp_dport, tcp_flags, tcp_ack, dan tcp_window. Dibawah akan dijelaskan mengenai nilai yang terkandung dalam field-field tesebut :

1. tcp_sport berisi informasi alamat port tcp yang digunakan oleh source ip-address selama melakukan koneksi melalui protokol tcp. Sedangkan field tcp_dport berisi informasi tujuan port tcp yang diakses oleh source ip-address selama koneksi melalui protokol tcp. Nilai bobot yang diberikan untuk kondisi abnormal yang berhubungan dengan ke dua field ini diwakili oleh variabel "port", sedangkan nilai bobot yang diberikan adalah :
   - port bernilai 0 jika kondisi dibawah tidak terpenuhi.
   - port bernilai 1 jika tcp_dport merupakan port 23.

   Setelah nilai bobot diberikan maka akan dimasukkan ke dalam field "port_status" di dalam tabel "status". Penggalan *listing code* untuk kondisi diatas adalah :

```
// mengambil nilai dari field tcp_sport dan tcp_dport //
strcpy(query,"select c.tcp_sport, c.tcp_dport, inet_ntoa(b.ip_src),
inet_ntoa(b.ip_dst) from iphdr b inner join tcphdr c on b.sid = c.sid and
b.cid = c.cid where inet_ntoa(ip_dst)='192.168.100.123'");
strcat(query," and inet_ntoa(ip_src)='");
strcat(query,ip); strcat(query,"'");
state=mysql_query(connection,query);
res=mysql_store_result(connection);
while((row=mysql_fetch_row(res))!=NULL)
```

```
{
      char banding1[100][100]; strcpy(banding1,row[0]);
      char banding2[100][100]; strcpy(banding2,row[1]);
      char port_telnet[2]; strcpy(port_telnet,"23");
      int compare3; compare3=strcmp(banding2,port_telnet);

      // cek apakah dst port == port 23 (telnet service) //
      if(compare3==0)
      {
        // beri bobot = 1 bila kondisi terpenuhi //
        session3=1;
        strcpy(query,"update status set port_status=1 where ip_src='");
        strcat(query,ip); strcat(query,"'");
        state=mysql_query(connection,query);
      }
}
```

2. tcp_flags berisi informasi mengenai flags yang aktif selama koneksi terjadi melalui protokol tcp. Nilai bobot yang diberikan terhadap kondisi yang berhubungan dengan nilai flags adalah sebagai berikut :

   - flags bernilai 0 jika semua kondisi dibawah tidak terpenuhi.
   - flags bernilai 1 jika nilai flags adalah "0" yaitu tidak ada flags yang aktif.
   - flags bernilai 2 jika flags yang aktif merupakan kombinasi flags FIN dan FIN saja.
   - flags bernilai 3 jika nilai flags adalah "0" dan flags yang aktif merupakan kombinasi dari flags FIN ataupun FIN saja.

   Setelah pemberian bobot maka nilai dari keadaan abnormal yang berhubungan dengan flags akan dimasukkan ke dalam tabel "status" dengan field bernama "flag_status". Penggalan *listing code* untuk kondisi ini adalah :

```
// pengambil nilai dari field tcp_flags //
state=mysql_query(connection,"select c.tcp_flags, inet_ntoa(b.ip_src),
inet_ntoa(b.ip_dst) from iphdr b inner join tcphdr c on b.sid = c.sid and
b.cid = c.cid where inet_ntoa(ip_dst)='192.168.100.123'");
int counter11=0; int counter21=0; int counter31=0; int counter41=0;
int counter51=0; int counter61=0; int counter71=0;
res=mysql_store_result(connection);
while((row=mysql_fetch_row(res))!=NULL)
{
  if(strcmp(row[2],"192.168.100.123")==0){
       if(strcmp(row[1],ip)==0){
            if(strcmp(row[0],"0")==0)
            {
              counter11++;
              if(counter11=3)
              {
                strcpy(query,"update status set flag_status=1 where ip_src='");
                strcat(query,ip); strcat(query,"'");
```

```
                state=mysql_query(connection,query);
              }
        }
        else if(strcmp(row[0],"1")==0)
        {
            counter21++;
            if(counter21=3)
            {
                strcpy(query,"update status set flag_status=2 where ip_sr='");
                strcat(query,ip); strcat(query,"'");
                state=mysql_query(connection,query);
            }
        }
        ----
        ----
    }
  }
}
```

3. tcp_ack berisi informasi mengenai besar nilai *acknowledgement number (ack)* yang masuk selama koneksi melalui protokol tcp. Untuk variabel ack tidak dikenakan nilai bobot, sebab merupakan nilai ack merupakan salah satu faktor untuk *anomaly detection* sehingga nilainya belum bisa dipastikan. Oleh karena itu dalam program ini yang dilakukan adalah mengambil nilai ack dan memasukkannya ke dalam field "tcp_ack" di tabel "status". Untuk penggalan listing code untuk kondisi diatas akan dijabarkan bersama dengan listing code nilai window dibawah.

4. tcp_window berisi informasi mengenai besar nilai *window* yang masuk selama terjadi koneksi melalui protokol tcp. Variabel window juga merupakan salah satu faktor untuk *anomaly detection* sehingga nilainya belum bisa dipastikan. Dalam program ini yang dilakukan adalah mengambil nilai window dan memasukkannya ke dalam field "tcp_window" di tabel "status", guna diproses lebih lanjut oleh program fuzzy.c. Penggalan listing code untuk kondisi diatas adalah :

```
// mengambil nilai ack dari field tcp_ack dan nilai window dari tcp_window //
strcpy(query,"select avg(c.tcp_ack),avg(c.tcp_win) from iphdr b inner join
tcphdr c on b.sid = c.sid and b.cid = c.cid where
inet_ntoa(ip_dst)='192.168.100.123' and inet_ntoa(ip_src)='");
```

```
strcat(query,ip); strcat(query,"'");
state=mysql_query(connection,query);
res=mysql_store_result(connection);
while((row=mysql_fetch_row(res))!=NULL)
{
  if(row[0]!=NULL) // ACK
  {
        char help2[20];
        strcpy(help2,row[0]);
        strcpy(query,"update status set tcp_ack='");
        strcat(query,help2); strcat(query,"'");
        strcat(query,"where ip_src='");
        strcat(query,ip); strcat(query,"'");
        state=mysql_query(connection,query);
  }
  if(row[1]!=NULL) // WINDOW
  {
        char help21[20];
        strcpy(help21,row[1]);
        strcpy(query,"update status set tcp_window='");
        strcat(query,help21); strcat(query,"'");
        strcat(query,"where ip_src='");
        strcat(query,ip); strcat(query,"'");
        state=mysql_query(connection,query);
  }
}
```

A.3. Tabel data

Tabel data berisi mengenai informasi data-data yang masuk ke dalam jaringan selama koneksi terjadi antara dua komputer atau lebih. Field yang digunakan nilainya adalah field *data_payload*. Nilai bobot yang diberikan untuk kondisi yang berhubungan dengan data_payload ini adalah :

- data_payload bernilai 0 jika semua kondisi dibawah tidak terpenuhi.
- data_payload bernilai 1 jika ditemukan data yang mengandung string “port” atau “open port”
- data_payload bernilai 2 jika ditemukan data yang mengandung string “incorrect” atau “login incorrect”.

  data_payload bernilai 3 jika ditemukan data yang mengandung string “port” atau “open port” dan mengandung string “incorrect” atau “login incorrect”.

Setelah pemberian bobot maka nilai dari keadaan abnormal yang berhubungan dengan data_payload akan dimasukkan ke dalam tabel status dengan field bernama “payload_status”. Penggalan listing code untuk kondisi ini adalah :

```
// mengambil nilai dari field data_payload dari tabel data //
strcpy(query,"select inet_ntoa(b.ip_src),inet_ntoa(b.ip_dst),a.data_payload from
data a inner join iphdr b on a.sid = b.sid and a.cid =b.cid where
inet_ntoa(ip_dst)='192.168.100.123' and inet_ntoa(ip_src)='");
strcat(query,ip); strcat(query,"'");
state=mysql_query(connection,query);
res=mysql_store_result(connection);
while((row=mysql_fetch_row(res))!=NULL)
{
      char compare1[100][100];
      strcpy(compare1,row[2]);
      char open_port1[18];
      strcpy(open_port1,"4F70656E20506F7274");
      char port_aja1[16];
      strcpy(port_aja1,"50206f2072207420");
      char wrong_login1[30];
      strcpy(wrong_login1,"4C6F67696E20696E636F7272656374");
      int posisi1; posisi1=strstr(compare1,open_port1);
      int posisi2; posisi2=strstr(compare1,wrong_login1);
      int posisi_bantu1;
      posisi_bantu1=strstr(compare1,port_aja1);
      // Contain 'Login Incorrect'
      if(posisi2!=0)
      {
            session1=1;
            strcpy(query,"update status set payl[illegible]atus=2 where ip_src='");
            strcat(query,ip); strcat(query,"'");
            state=mysql_query(connection,query);
      }
      // Contain 'Open Port' or 'Port'
      else if((posisi1!=0)||(posisi_bantu1!=0))
      {
            session2=1;
            strcpy(query,"update status set payload_status=1 where ip_src='");
            strcat(query,ip); strcat(query,"'");
            state=mysql_query(connection,query);
      }
}
```

A.4. Tabel icmphdr

Tabel icmphdr berisi nilai-nilai yang berhubungan dengan aktivitas jaringan yang menggunakan protokol icmp. Dalam sistem ini penggunaan tabel icmphdr hanya didasarkan atas banyak atau tidaknya koneksi yang menggunakan protokol icmp, sedangkan nilai-nilai yang terkandung di dalam field-fieldnya diabaikan. Nilai bobot yang diberikan untuk kondisi yang berhubungan dengan tabel icmphdr ini adalah :

- icmp bernilai 0 jika jumlah total protokol icmp yang aktif <= 70.
- icmp bernilai 1 jika jumlah total protokol icmp yang aktif > 70.

Setelah pemberian bobot maka nilai dari keadaan abnormal yang berhubungan dengan tabel icmphdr akan dimasukkan ke dalam tabel status dengan field bernama "icmp_status". Penggalan listing code untuk kondisi ini adalah :

```
// melihat isi dari tabel icmphdr //
strcpy(query,"select c.* from iphdr b inner join icmphdr c on b.sid =
b.cid = c.cid where inet_ntoa(ip_dst)='192.168.100.123' and inet_ntoa(ip_src)=
strcat(query,ip); strcat(query,"'"); state=mysql_query(connection,
res=mysql_store_result(connection);
if((row=mysql_fetch_row(res))==NULL)
{  printf("\t\t\t      Null Data on Tabel ICMP\n\n");  }
else
{   if(mysql_num_rows(res)<=70)
    {  printf("\t\t\t  Nothing Suspicious on ICMP\n\n");
       strcpy(query,"update status set icmp_status=0 where
       strcat(query,ip); strcat(query,"'"); state=mysql_query(
    }
    else { strcpy(query,"update status set icmp_status
---
---
}
```

A.5. Tabel udphdr

Tabel udphdr berisi nilai-nilai yang berhubungan dengan aktivitas jaringan yang menggunakan protokol udp. Dalam sistem ini penggunaan tabel udphdr sama dengan tabel icmphdr diatas, yaitu hanya didasarkan atas banyak atau tidaknya koneksi yang menggunakan protokol icmp, sedangkan nilai-nilai yang terkandung di dalam field-fieldnya diabaikan. Nilai bobot yang diberikan untuk kondisi yang berhubungan dengan tabel udphdr ini adalah :

- udp bernilai 0 jika jumlah total protokol icmp yang aktif <= 50.
- udp bernilai 1 jika jumlah total protokol icmp yang aktif > 50.

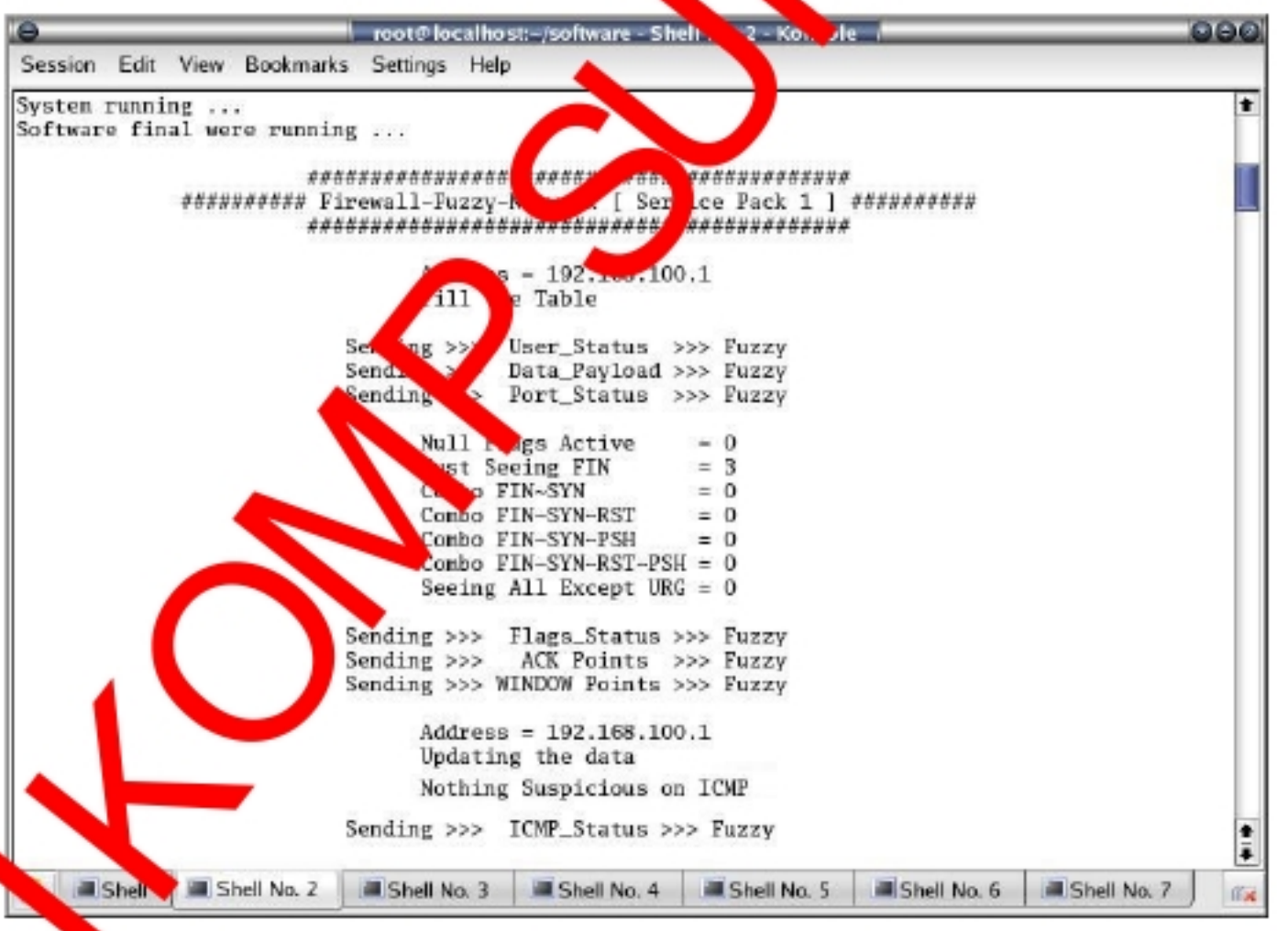
Setelah pemberian bobot maka nilai dari keadaan abnormal yang berhubungan dengan tabel udphdr akan dimasukkan ke dalam tabel status dengan field bernama "udp_status". Penggalan listing code untuk kondisi ini adalah :

```
// melihat isi dari tabel udphdr //
strcpy(query,"select c.* from iphdr b inner join udphdr c on b.sid = c.sid and
b.cid = c.cid where inet_ntoa(ip_dst)='192.168.100.123' and inet_ntoa(ip_src)='");
strcat(query,ip);  strcat(query,"'");  state=mysql_query(connection,query);
res=mysql_store_result(connection);
if((row=mysql_fetch_row(res))==NULL)
{ printf("\t\t\t      Null Data on Tabel UDP\n\n"); }
else
```

```
{ if(mysql_num_rows(res)<=50)
  { printf("\t\t\t     Nothing Suspicious on UDP\n\n");
      strcpy(query,"update status set udp_status=0 where ip_src='");
      strcat(query,ip); strcat(query,"'"); state=mysql_query(connection,query);
  }
  else { strcpy(query,"update status set udp_status=1 where ip_src='"); }
---
---
}
```

A.6. Tabel status

Tabel status merupakan tabel untuk menampung nilai ip_src, ip_dst, tcp_sport, tcp_dport, tcp_flags, tcp_ack, tcp_window, data payload. Tabel status merupakan kumpulan nilai yang akan dijadikan masukan bagi program fuzzy.c. Visualisasi software final.c ditunjukkan pada gambar 3.18 dibawah ini :



Gambar 3.18 Visualisasi software final.c

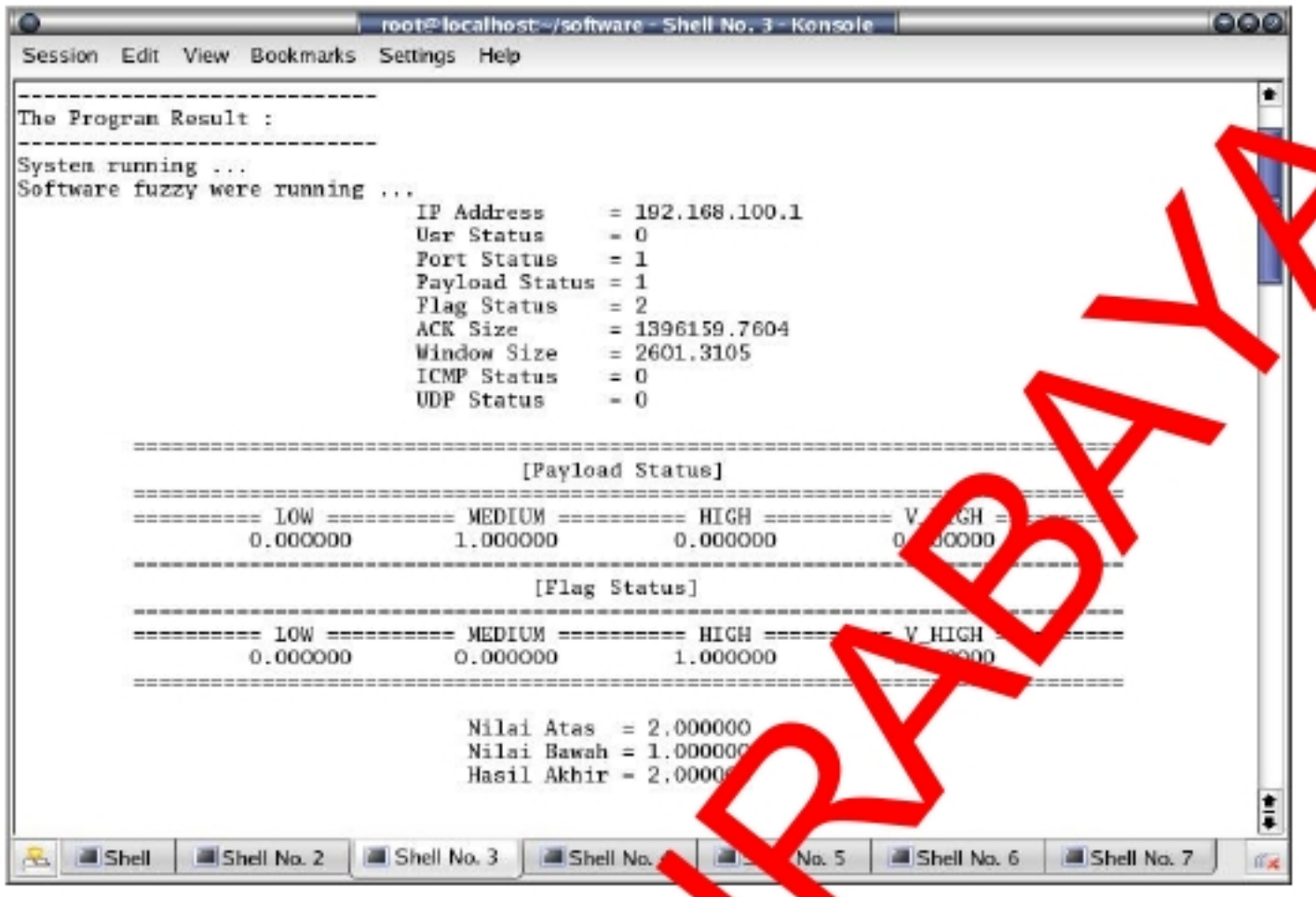
B. fuzzy.c

Program fuzzy.c berfungsi untuk melakukan perhitungan fuzzy terhadap nilai-nilai yang terdapat pada tabel "status", mulai dari inisialisasi fungsi keanggotaan dan perhitungan defuzzifikasi, kemudian nilai-nilai hasil defuzzifikasi tersebut dimasukkan ke dalam tabel "fuzzy".

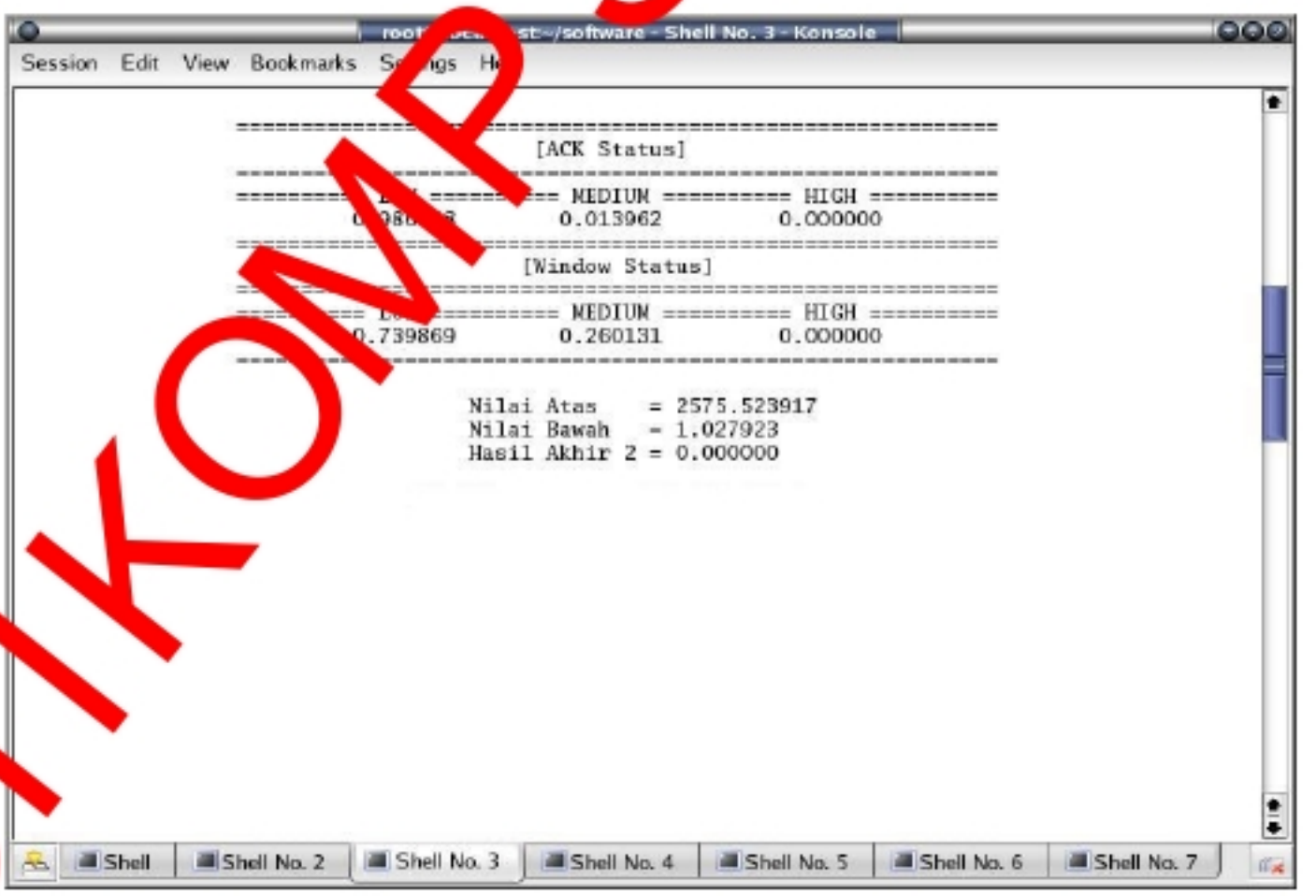
Penggalan listing code untuk perhitungan fuzzy akan dijabarkan sebagai berikut :

```
// mengambil inputan fuzzy dari tabel status digabungkan dengan tabel fuzzy //
state=mysql_query(connection,"select a.*, b.usr_port, b.pay_flag,
b.ack_window, b.icmp_udp from fuzzy b inner join status a on b.ip_src =
a.ip_src");

// inisialisasi batas dan nilai tengah inputan fungsi keanggotaan segitiga //
a1=-1; b1=0; c1=1; a2=0;  b2=1; c2=2; a3=1;  b3=2; c3=3; a4=2;  b4=3; c4=4;
bo1=0; bo2=1; bo3=2; bo4=3;

// hitung fungsi keanggotaan segitiga -> z=max(min(...),0) //
// 1. cari nilai (min) terlebih dahuku -> res=min(...) //
hasil1=((x1-a1)/(b1-a1));
hasil2=((c1-x1)/(c1-b1));
if(hasil1<=hasil2){min1=hasil1;}
else if(hasil2<=hasil1){min1=hasil2;}
-
-
// 2. cari nilai (max) -> max(res_min) [nilai bantu=0] //
if(min1>=bantu1){low=min1;}
else if(bantu1>=min1){low=bantu1;}
if(min2>=bantu2){med=min2;}
else if(bantu2>=min2){med=bantu2;}
-
-
// cari tinggi fungsi segitiga hasil inferensi fuzzy //
if(low<=low1){w1=low;}
else if(low1<=low){w1=low1;}
if(low<=med1){w2=low;}
else if(med1<=low){w2=med1;}
if(low<=high1){w3=low;}
else if(high1<=low){w3=high1;}
if(low<=vhigh1){w4=low;}
else if(vhigh1<=low){w4=vhigh1;}
-
-
// defuzzifikasi dengan metode COA //
atas=(bo1*w1)+(bo2*w2)+(bo2*w3)+(bo3*w4)+(bo2*w5)+(bo2*w6)+(bo3*w7)+(bo4*w8)+(bo2
*w9)+(bo3*w10)+(bo3*w11)+(bo4*w12)+(bo4*w13)+(bo4*w14)+(bo4*w15)+(bo4*w16));
bawah=w1+w2+w3+w4+w5+w6+w7+w8+w9+w10+w11+w12+w13+w14+w15+w16;
CA=fungsi(atas,bawah);

// memasukkan nilai hasil defuzzifikasi ke tabel fuzzy //
char usr_port[100]; snprintf(usr_port,100,"%f\n",CA);
strcpy(query,"update fuzzy set usr_port='"); strcat(query,usr_port);
strcat(query,"'"); strcat(query,"where ip_src='"); strcat(query,ip);
strcat(query,"'");
state=mysql_query(connection,query);
```

Sedangkan visualisasi dari software fuzzy.c ditunjukkan pada gambar 3.19 dan 3.20 dibawah ini :



Gambar 3.19 Visualisasi software fuzzy.c (bagian 1)



Gambar 3.20 Visualisasi software fuzzy.c (bagian 2)

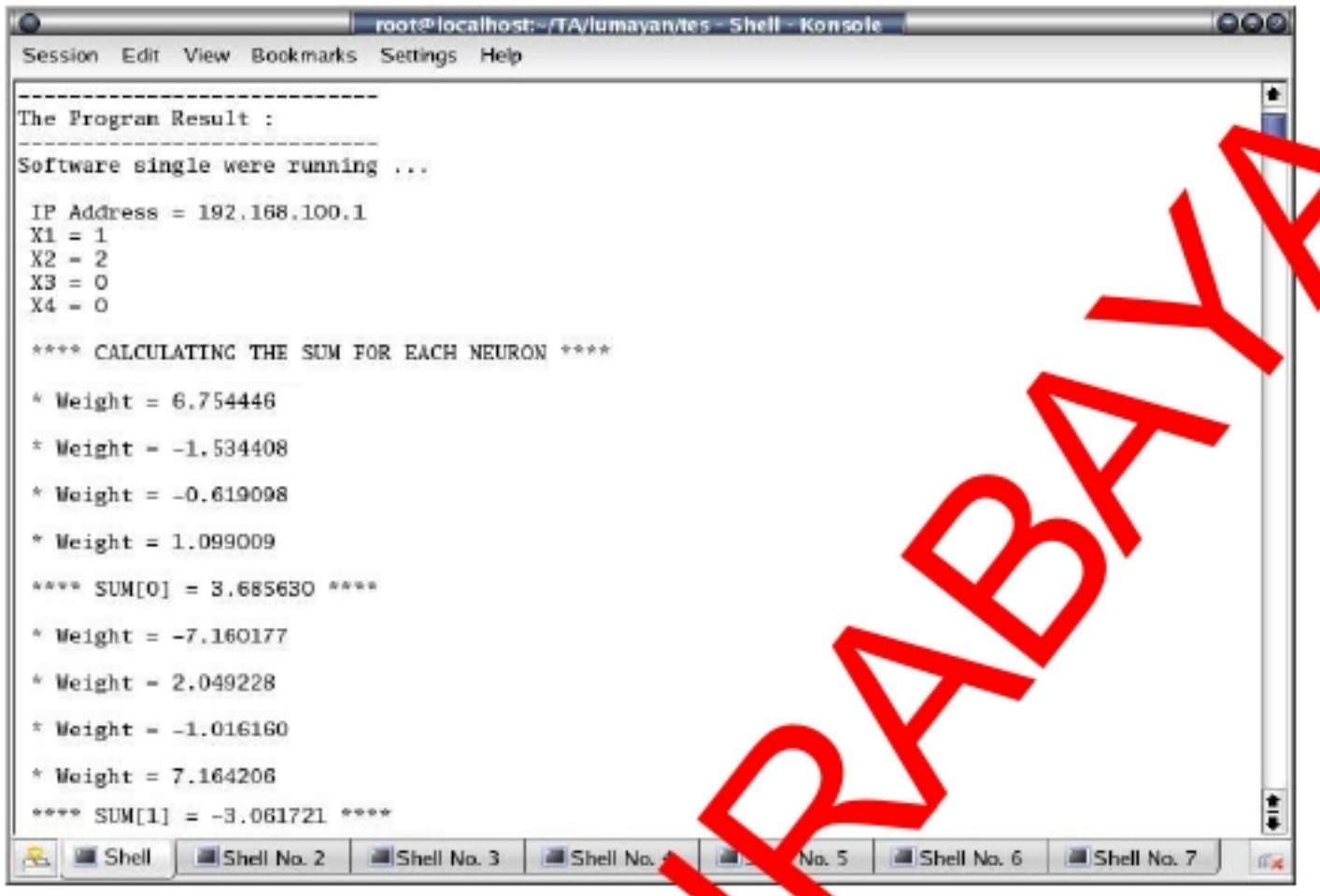
C. single.c

Program single.c berfungsi untuk melakukan proses *testing neural network* dari berbagai macam kondisi hasil perhitungan fuzzy yang bera a pa a tabel "fuzzy", sehingga bisa dikelompokkan ke dalam empat macam k ndis yang terjadi, yaitu normal, *low attack*, *medium attack* dan *high attac* . Proses testing hanya melibatkan perhitungan feed forward saja.

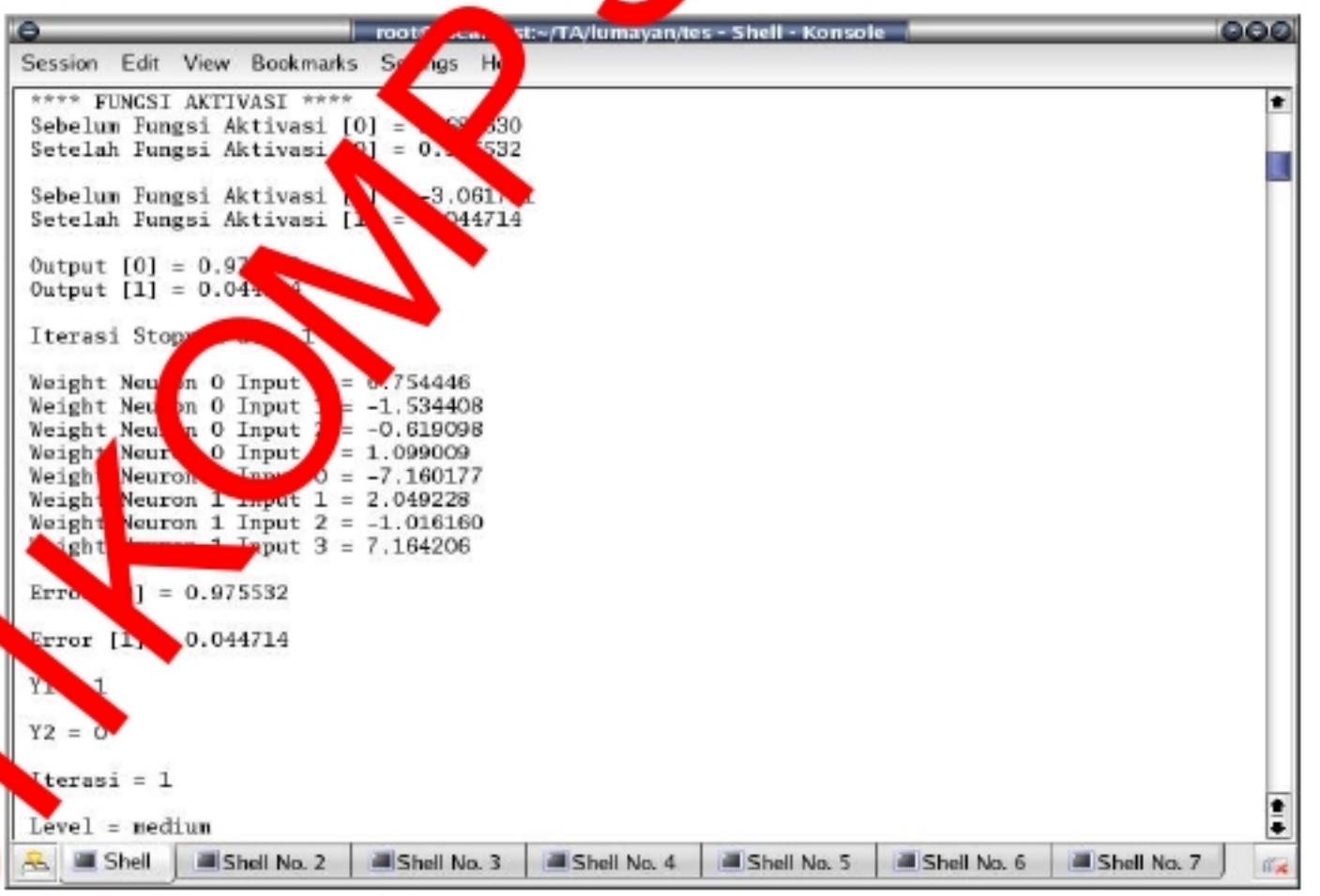
Keluaran dari proses pengujian neural networ ak n di asukkan ke dalam tabel "jst". Dibawah ini akan dicantumkan pe gg dari sting code dari proses inisialisasi masukan dan testing, yaitu :

```
// mengambil data dari tabel fuzzy untuk dija kan  nput //
state=mysql_query(connection,"select usr_p rt, p load_flag, ack_window, icmp_udp
from fuzzy where status='complete' limit 1");
res=mysql_store_result(connection);

// ambil nilai weight hasil training t akhir di abel bobot //
strcpy(query,"select * from bobot2 e put='");
strcat(query,gabung2);
strcat(query,"'");
state=mysql_query(connection,query
res=mysql_store_result(connection);

// lakukan perhitungan fee orw d //
// perhitungan "sum" untu masi - masing neuron //
for(neuron_num=0; neuron m<2 neuron_num++)
{
    sum[neuron_num] =
    for(neuron_input=0 uron_i ut<4; neuron_input++)
    {
          product = input neu n_input]*weight[neuron_num][neuron_input];
          sum[ne um] = sum[neuron_num] + product;
    }
          su m] = 0 + sum[neuron_num];
          p intf(" M[ ] = %f\n",neuron_num,sum[neuron_num]);
}

// mas k ke fungsi tivasi //
sum1[n uron_n ];
act_ne ron1 = 1 / (1+exp(-1*sum[neuron_num]));
su 1[ne uron_num] = act_neuron1;

// has akhir --> //
or(neu num=0; neuron_num<2; neuron_num++)
{
  i um1[neuron_num] < 0.5)
  {       hasil[neuron_num] = 0;       }
  else
  {       hasil[neuron_num] = 1;       }
}
printf(" Output Neuron [0] = %d\n",hasil[0]);
printf(" Output Neuron [1] = %d\n",hasil[1]);
```

Sedangkan visualisasi dari software single.c ditunjukkan pada gambar 3.21 dan 3.22 dibawah ini :



Gambar 3.21 Visualisasi software single.c (bagian 1)



Gambar 3.22 Visualisasi software single.c (bagian 2)

D. blocking.c

Program blocking.c berfungsi untuk membaca hasil keluaran dari program single.c yang berada di dalam tabel "jst". Setelah itu akan dilakukan reaksi atas keluaran yang terjadi, yaitu melakukan *blocking ip-address* berdasarkan level kondisi abnormal yang terjadi. Tiap level kondisi abnormal mempunyai tipe blocking yang berbeda, antara lain :

- Kondisi normal, berarti ip-address tidak akan diblok.
- Kondisi *low*, berarti ip-address akan diblok selama 1 jam.
- Kondisi *medium*, berarti ip-address akan diblok selama 1 hari.
- Kondisi *high*, berarti ip-address akan diblok selama 3 hari.

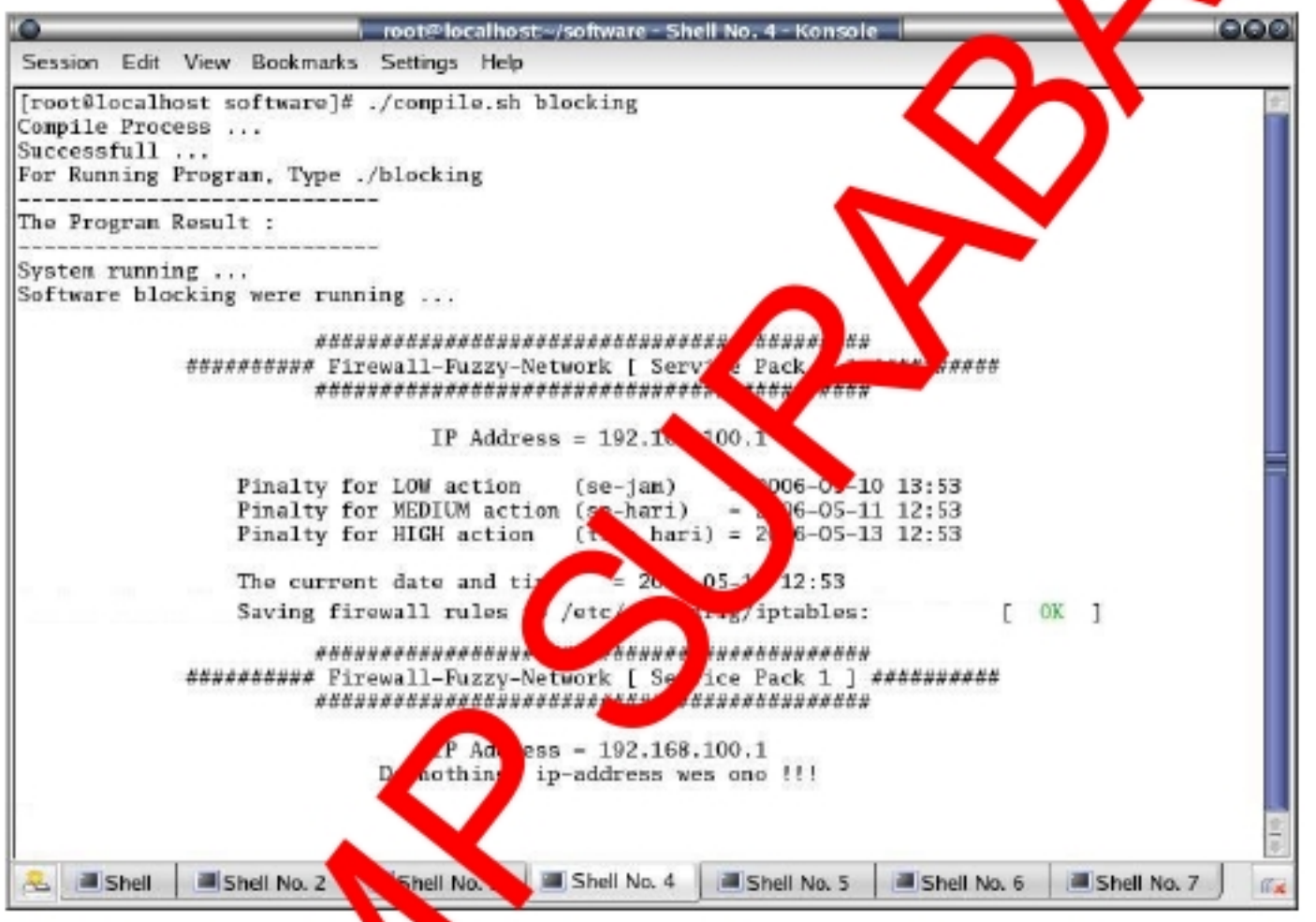
Keluaran dari program blocking.c untuk kondisi yang dikategorikan serangan akan dilakukan penyimpanan data berupa ip-address yang diblok beserta waktu mulai diblok sampai dengan waktu blocking dibebaskan, sedangkan untuk kondisi normal tidak dilakukan penyimpanan data.

Listing code untuk kondisi diatas adalah sebagai berikut :

```
// mengambil data dari tabel jst //
state=mysql_query(connection,"select * from jst");
res=mysql_store_result(connection);
char restart[24]; strcpy(restart,"service iptables restart");
char save[2?]; strcpy(save,"service iptables save");
char ip[15]; strcpy(ip,row[0]);
char output[10]; strcpy(output,row[4]);
char server[15]; strcpy(server,"192.168.100.123");

// fungsi mengambil jam waktu //
time_t tval; struct tm *now;
char bu[?]; tval = time(NULL);
now = localtime(&tval);

char waktu_sejam[100];snprintf(waktu_sejam,100,"%04d-%02d-%02d %02d:%02d",now-
>tm_year+1900,now->tm_mon+1,now->tm_mday,now->tm_hour+1,now->tm_min);
char time_block1[120]; strcpy(time_block1,waktu_sejam);

// memasukkan ip-address penyerang ke dalam sintak iptables //
char sintak[108]; strcpy(sintak,"INPUT -s "); strcat(sintak,ip);
strcat(sintak," -d "); strcat(sintak,server); strcat(sintak," -j DROP");
char append[120]; strcpy(append,"iptables -A "); strcat(append,sintak);

// cek nilai output dari tabel jst --> sesuaikan dengan blocking time //
// lalu simpan ke tabel blocking //
if(strcmp(output,"01")==0) {
strcpy(query,"insert into blocking (ip_src,date_awal,date_akhir,sintak)
values ('"); strcat(query,ip); strcat(query,"',"); strcat(query,"'");
```

```
strcat(query,time_masuk); strcat(query,"',"); strcat(query,"'");
strcat(query,time_block1); strcat(query,"',"); strcat(query,"'");
strcat(query,sintak);  strcat(query,"')"); state=mysql_query(connection,query);
system(append);
system(save);
system(restart);
```

Sedangkan visualisasi dari software blocking.c ditunjukkan pada gambar 3.23 dibawah ini :



Gambar 3.23 Visualisasi software blocking.c

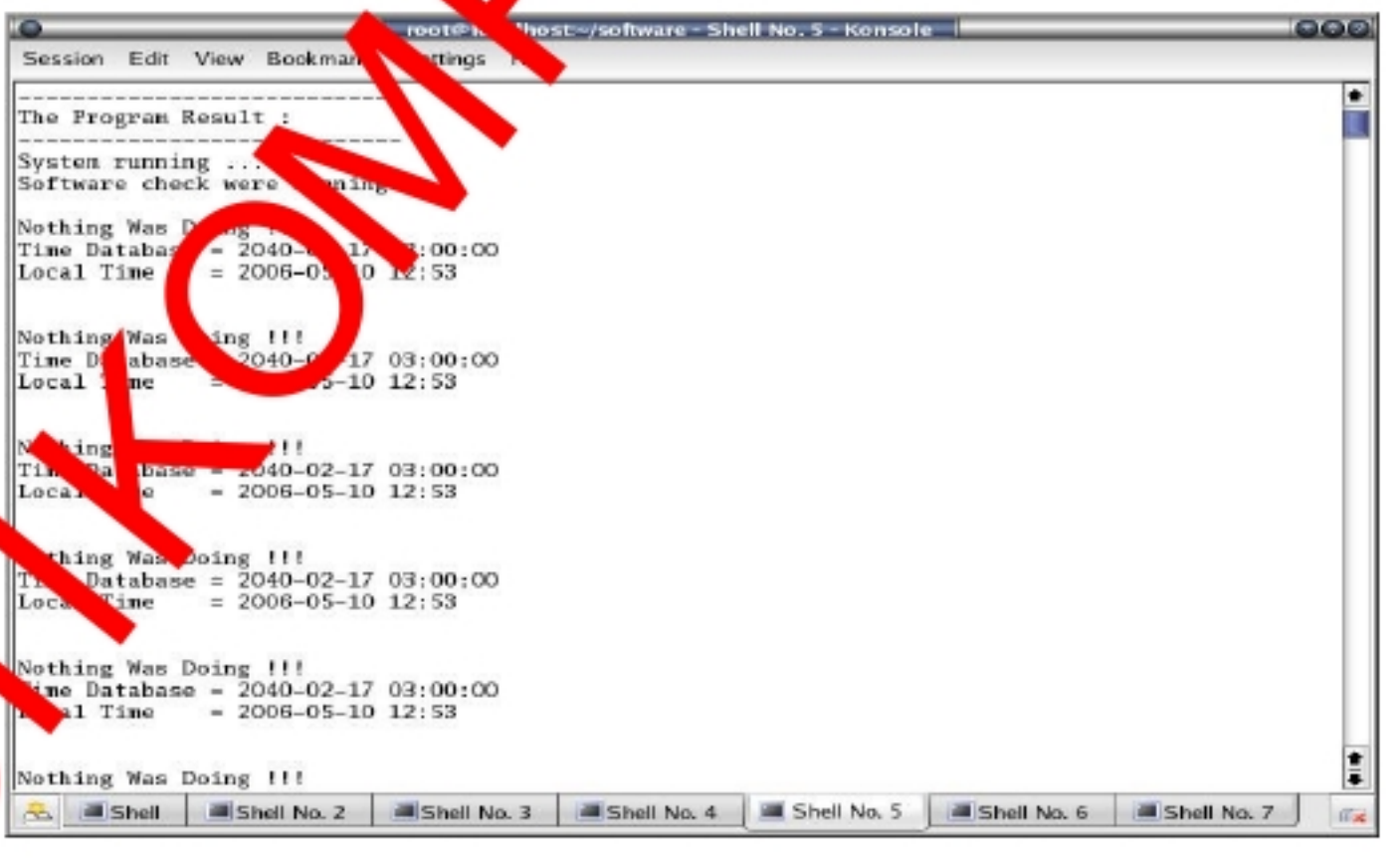
E. check.c

Program check.c berfungsi untuk membaca isi dari tabel "blocking" untuk membaca batas waktu dari ip-address yang sedang diblok. Kemudian program akan membaca waktu dari jam sistem saat itu (*localtime*) untuk dibandingkan, apabila bernilai sama atau lebih besar dari nilai jam sistem, maka akan dilakukan penghapusan secara otomatis terhadap *rule iptables* sesuai dengan

ip-address yang sedang diblok. Listing code dari kondisi diatas akan ditunjukkan dibawah ini :

```
// mengambil data dari tabel blocking //
state=mysql_query(connection,"select * from blocking");
res=mysql_store_result(connection);

// mengambil waktu saat ini dari localtime //
time_t tval; struct tm *now; char buf[BUFSIZE]; tval = time(NULL);
now = localtime(&tval);
char waktu_lokal[100];
snprintf(waktu_lokal,100,"%04d-%02d-%02d %02d:%02d:%02d",now->tm_ye  +19 ,
>tm_mon+1,now->tm_mday,now->tm_hour,now->tm_min);
char time_lokal[120]; strcpy(time_lokal,waktu_lokal); puts(b
char date_akhir[20]; strcpy(date_akhir,row[2]);

// melakukan pengecekan apakah waktu di tabel == waktu saat
int compare;
compare=strcmp(time_lokal,date_akhir);
if(compare==0)
{
    strcpy(query,"select * from blocking where da akhi );
    strcat(query,time_lokal); strcat(query,"'");
    state=mysql_query(connection,query); char ntak abel 08];
    strcpy(sintak_tabel,row[3]); printf("Sint = %  tak_tabel);
    char sintak[120];  strcpy(sintak,"iptables
    strcat(sintak,sintak_tabel);
    printf("The Iptables Syntax          = %s \n",  tak);
    strcpy(query,"delete from blocking where da akhir='");
    strcat(query,time_lokal); strcat( ery,"'");
    state=mysql_query(connection,que
    system(sintak); system(save);  stem(r
}
```

Sedangkan visualisasi dari software ch k.c ditunjukkan pada gambar 3.24 dibawah ini :



Gambar 3.24 Visualisasi software check.c

F. learning.c

Sebelum masuk ke tahap *testing neural network,* maka suatu pasangan pola masukan dan target harus di-*training* (dilatihkan) terlebih dahulu, sampai mencapai keluaran yang diinginkan. Program learning.c ini berfungsi untuk melakukan proses pelatihan/pembelajaran neural network dengan menggunakan metode backpropagation. Dalam tahap pembelajaran (*learning algorithm*) ini melibatkan beberapa tahap yaitu perhitungan feed forward, perhitungan error dengan backpropagation dan penyesuaian bobot (*update weight*). Pasangan pola masukan dan pola target disimpan di tabel training, selama keluaran belum mencapai hasil yang diinginkan maka bobot yang digunakan untuk tahap pembelajaran ini diperbaharui (*update*) dan disimpan di dalam tabel bobot. Proses pembelajaran berhenti apabila nilai error dibawah atau sama dengan nilai error minimum, sedangkan untuk pembatasan agar proses pembelajaran tidak terjebak oleh *looping* yang lama, maka diberikan batasan jumlah iterasi sebanyak 20.000. Bobot terakhir hasil pembelajaran akan menjadi bobot untuk tahap pengujian. Listing code dari kondisi diatas akan ditunjukkan dibawah ini :

```
// mengambil data dari tabel learning untuk dijadikan input latihan //
state=mysql_query(connection,"select * from training where status='complete' limit
1");
res=mysql_store_result(connection);


// inisialisasi bobot random untuk latihan //
for(neuron_num=0; neuron_num<2; neuron_num++)
{
  for(neuron_input=0; neuron_input<4; neuron_input++)
  {
          int bantu1 = (5.0*rand()/(RAND_MAX+1.0));
          int bantu2 = (-5.0*rand()/(RAND_MAX+1.0));
          int bantu3 = bantu1 + bantu2;

          double bantu4 = bantu3 / 10.0;
          weight[neuron_num][neuron_input]=bantu4;
  }
}

// selama kondisi = false //
while(train_flag==0)
{
  iterasi++;
```

```
// hitung net input --> Xj = Zigma[i] ( Yi*Wji ) //
for(neuron_num=0; neuron_num<2; neuron_num++)
{
        sum[neuron_num] = 0;
        for(neuron_input=0; neuron_input<4; neuron_input++)
        {
              product = input[neuron_input]*weight[neuron_num][neuron_input];
              sum[neuron_num] = sum[neuron_num] + product;
        }
        sum[neuron_num] = 0 + sum[neuron_num];
}

// masuk ke fungsi aktivasi --> Yj = 1 / (1 + exp[-Xj]) //
for(neuron_num=0; neuron_num<2; neuron_num++)
{
        sum1[neuron_num] = 0;
        act_neuron1 = 1 / (1+exp(-1*sum[neuron_num]));
        sum1[neuron_num] = act_neuron1;
}

// hitung error yang terjadi --> E = 0.5 * sqr(t[j]-Y[j]) //
eror1 = 0;    eror1 = ((0-sum1[0])*(0-sum1[0]));    e    1 = 0.5    ror1;
error_num[0] = eror1;
eror2 = 0;    eror2 = ((0-sum1[1])*(0-sum1[1]));    er   2 = 0.5    eror2;
error_num[1] = eror2;

// selama >= error_minimum, masuk ke backpro  gati  //
if((error_num[0] >= error_max) || (error_nu  1] >      max))
{       error_flag = 0;       }

// bacpropagation --> 'E/'Yj = t[j] - Y[j]    targ  - output actual //
if(error_flag == 0)
{       cek[0] = 0 - sum1[0];          k[1] = 0    sum1[1];          }

// update new weight //
for(neuron_num=0; neuron_num<2;  euron  )
{
  if(error_num[neuron_num]>=error_max)
  {
    for(neuron_input=0; n     input<4; neuron_input++)
    {
        wt =  weight[ne   on_nu  [neuron_input] +
              ((b*cek[ne    n    m]) *input[neuron_input]);

        weight[neuro      m][ne    on_input] = wt;
    }
  }
}

// selama e      > error_minimum atau iterasi != 20000 //
// then t    us la    ka  pembelajaran //
// simpa  nilai bo    ot di tabel bobot //
```