BAB 5
REKABENTUK DAN PENDEKATAN SISTEM REDALERT

5.0 REKABENTUK SISTEM REDALERT


Aplikasi utama dalam sistem RedAlert adalah RedAlertIDS yang merupakan program induk. Ia terdiri daripada 9 fungsi dan bertindak secara masa nyata. Berikut adalah kumpulan fungsian dalam program RedAlertIDS.

i. redalert_event()

Fungsi yang menyemak kandungan pangkalan data dan menentukan sama ada terdapat sebarang pencerobohan.

ii. redalert_check()

Membandingkan kejadian pencerobohan dengan pangkalan data untuk memastikan tindakan selanjutnya ke atas penceroboh terbabit.
iii. redalert_policy()
Fungsi yang membentuk polisi baru bagi setiap pencerobohan yang berlaku. Bergantung kepada konfigurasi yang ditetapkan oleh pentadbirdalam pangkalan data.

iv. redalert_store()
Fungsi yang menyimpan tindakan yang dikenakan ke atas perubahan polisi keselamatan rangkaian, setiap polisi yang dibuat akan disimpan dalam pangkalan data.

v. redalert_clear()
Fungsi yang mengemaskan polisi yang telah dibentuk. Bergantung kepada pangkalan data sama ada polisi perlu dibuang atau dikekalkan.

vi. redalert_signal()
Fungsi yang menghantar transmisi kepada semua hos melalui transmisi multicast dan unicast bagi subnet berbeza.1

vii. redalert_getdata_conf()
Fungsi yang mendapatkan konfigurasi bagi proses perlaksanaan program termasuklah IP bagi multicast, unicast dan MySQL. Ia termasuk beberapa konfigurasi penting seperti kunci penyulitan, port, pengguna dan katalaluan bagi pangkalan data MySQL.
viii. redalert_unicast()

    Sub fungsian bagi redalert_signal bagi menghantar transmisi unicast.

ix. timered()

    Fungsi yang memperolehi tempoh masa polisi dalam firewall. Bergantung kepada pentadbir untuk menjangka tempoh masa.


5.1 HIERARKI ALGORITMA APLIKASI REDALERTIDS

Hierarki algoritma aplikasi RedAlertIDS merupakan unjuran sub program yang saling kait yang membentuk sistem RedAlertIDS. Berikut adalah algoritma bagi aplikasi RedAlertIDS:

MULA PROGRAM (1)
GET NilaiID
IF Null(NilaiID) THEN
    EXIT()
ELSE
    WHILE (1)
        SLEEP(1);
        SUBPROGRAM();
        redalert_clear();
    END WHILE
END IF
TAMAT PROGRAM (1)
Hierarki dalam aplikasi RedAlertIDS terbentuk daripada algoritma ringkas di atas di mana kumpulan fungsian yang berkaitan dijalankan dalam SUBPROGRAM(). Pecahan dalam sub program terbahagi kepada beberapa proses lain yang menjalankan fungsii masing-masing. NilaiID merujuk kepada nilai terakhir yang diproses dalam aplikasi RedAlertIDS sebelumnya. Nilai tersebut digunakan untuk mendapatkan titik akhir sebelum aplikasi dimatikan sebelumnya. Aplikasi juga menjalankan proses SLEEP(1) bagi memastikan kadar penggunaan CPU tidak digunakan 100%. Jika tiada penggunaan SLEEP(1) dalam aplikasi, ia akan menggunakan CPU dengan kadar yang tinggi. Penggunaan SLEEP(1) melambatkan seketika (<1 saat). Ini membolehkan CPU beroberoperasi dengan baik.

Dalam sub program aplikasi ini, terdapat beberapa proses yang saling berkaitan. Lanjutan daripada algoritma di atas adalah:

```
MULA SUBPROGRAM
IF redalert_event(1) THEN
   IF redalert_check(1) THEN
      IF redalert_polisi(0) THEN
         redalert_store();
      END IF
   END IF
   NilaiID++;
END IF
```

Sub program akan menjalankan beberapa fungsii berkaitan seperti yang dinyatakan. Sub program merupakan fungsian utama yang berjalan secara berterusan sehingga sistem ditamatkan. Ianya merupakan gelung tanpa putus menggunakan while(1) dalam program utama. Fungsian redalert_event() dijalankan terlebih dahulu, jika fungsii memulangkan
nilai 1 maka fungsi redalert_check() akan dilaksanakan. Fungsi redalert_polisi() akan berjalan sekiranya redalert_check() memulangkan nilai 1 juga. Seterusnya redalert_store() akan terlaksana jika redalert_polisi() memulangkan nilai 0. Perlu dinyatakan juga nilaiID akan bertambah 1 apabila proses selesai. Ini membolehkan proses menyemak dan mengemaskini data pencerobohan yang berlaku kemudian.

5.2 FUNGSIAN APLIKASI REDALERTIDS

Perlakuan dalam fungsian merujuk kepada perjalanan sistem utama. Berikut adalah perkaitan antara sub program dalam aplikasi RedAlertIDS.

```
MULA redalert_event()
GET event
    IF Null(event) THEN
        Nothing;
    ELSE
        GET data;
        X=1;
    END IF
RETURN X;
TAMAT redalert_event()
```

Sub program ini akan mendapatkan kejadian yang berlaku dalam rangkaian. Data yang tersimpan dalam pangkalan data akan dibaca oleh sub program ini sebelum proses seterusnya dijalankan. Jika tiada sebarang kejadian, aplikasi akan terus berjalan. Jika terdapat kejadian pencerobohan, nilai 1 akan dipulangkan ke program utama untuk proses seterusnya iaitu redalert_check().
MULA redalert_check()
GET Signature ID
    IF SignatureID THEN
    GET RedAlertID
    IF RedAlertID==SignatureID THEN
    STORE data;
    X=1;
    END IF
    END IF
RETURN X;
TAMAT redalert_check()

Fungsi red_alert_check() berfungsi dalam mengenal pasti pencerobohan yang berlaku, sama ada ia merupakan pencerobohan yang hendak dikawal atau sebaliknya. Bentuk pencerobohan dibandingkan dengan pangkalan data untuk RedAlert. Jika sama, data disimpan dan nilai 1 dipulangkan kepada program utama untuk proses seterusnya. Dua proses panggilan pangkalan data terlibat iaitu:

   i. Mendapatkan signature ID berkaitan daripada pangkalan data signature.
   ii. Membandingkan signature ID dengan pangkalan data redalert bagi memastikan bentuk pencerobohan yang perlu dikawal atau tidak.

Proses seterusnya jika kejadian perlu dikawal ialah redalert_polisi. Pada redalert_polisi, proses penghasilan polisi dijalankan. Beberapa kriteria diperhatikan dalam menghasilkan polisi termasuklah:

   i. Alamat sumber penceroboh dan destinasi pencerobohan.
   ii. Jenis protokol rangkaian sama ada TCP,UDP atau ICMP.
   iii. Port sumber dan destinasi bagi protokol TCP dan UDP. ICMP tidak memerlukan port.
   iv. Tindakan rantaian firewall. INPUT, OUTPUT dan FORWARD.
Setelah polisi dipertimbangkan, proses seterusnya ialah proses redalert_store().

```
MULA redalert_store()
    MASUKKAN data ke pangkalan data;
    redalert_signal();
TAMAT redalert_store()
```

Fungsi redalert_store menyimpan semua maklumat yang akan dilaksanakan termasuklah masa yang diperlukan untuk tempoh firewall aktif. Fungsi timedel() digunakan bagi menghasilkan masa yang berkaitan mengikut pangkalan data pentadbir.

Fungsi redalert_signal() dalam redalert_store merujuk kepada proses penghantaran polisi yang dibentuk. Berikut adalah butiran mengenai fungsi redalert_signal.

i. Menjalankan proses penyulitan XOR blok.

ii. Berperanan menghantar polisi dan arahan menggunakan transmisi multicast.

iii. Terdapat dua jenis transmisi iaitu dalam protokol multicast dan unicast, maka terdapat satu sub fungsi iaitu redalert_unicast.

iv. Fungsi redalert_unicast menghantar menggunakan transmisi unicast.

Algoritma penyulitan XOR blok.

```
for(enc=0;enc<strlen(source);enc++)
{
    if(enc==0)
        memset(dest,"0",MAX_MSG);
        dest[enc] = source[enc] ^ key[y];
        y++;
    if(y==strlen(key))
        y=0;
}
```

XOR blok merujuk kepada setiap huruf dalam polisi ditukar dengan proses XOR dengan kunci yang dinyatakan oleh pentadbir sistem. Proses penyulitan dan nyah penyulitan
 adalah sama. Penggunaan kunci bagi proses ini adalah simetri iaitu menggunakan kunci yang sama. Simbol ∧ merujuk kepada XOR dalam aturcara C.

Fungsi yang terakhir merupakan redalert_clear() yang menamatkan polisi sedia ada dalam firewall mengikut aturan masa dalam pangkalan data. Fungsi ini juga menggunakan fungsi redalert_signal() sebagaimana dalam redalert_store().

Fungsi lain yang tidak memainkan peranan secara langsung tetapi diperlukan untuk melengkapi kitaran RedAlertIDS adalah redalert_getdata_conf, timered, mysql_connected, dan mysql_disconnected. Fungsian berikut menjalankan fungsi seperti yang dinyatakan sebelum ini.
5.3 REKABENTUK INTEGRASI PANGKALAN DATA

Integrasi pangkalan data merupakan penggabungan pangkalan data sedia ada dengan pangkalan data yang dicipta. Pangkalan data sedia wujud dalam sistem pengesan pencerobohan Snort menggunakan pangkalan data MySQL yang mempunyai beberapa jadual yang berkaitan. Aplikasi RedAlert menambah 2 lagi jadual untuk memberi satu bentuk hubungan yang boleh dikaitkan antara satu sama lain. Namun demikian hubungan antara jadual sedia ada dengan jadual baru tidak banyak memberi kesan negatif kepada sistem malahan memberi kesan yang positif. Berikut adalah ringkasan mengenai pangkalan data yang digunakan:

Jadual 5.1 : Deskripsi dalam pangkalan data

<table>
<thead>
<tr>
<th>Jadual</th>
<th>Komponen</th>
<th>Deskripsi</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheme</td>
<td>Snort</td>
<td>Maklumat mengenai pangkalan data.</td>
</tr>
<tr>
<td>sensor</td>
<td>Snort</td>
<td>Nama pengesan.</td>
</tr>
<tr>
<td>event</td>
<td>Snort</td>
<td>Meta data mengenai pencerobohan yang dikesan.</td>
</tr>
<tr>
<td>signature</td>
<td>Snort</td>
<td>Senarai bentuk pencerobohan, prioriti, dan rujukan ID.</td>
</tr>
<tr>
<td>sig_refrence</td>
<td>Snort</td>
<td>Rujukan tambahan bagi bentuk pencerobohan.</td>
</tr>
<tr>
<td>reference</td>
<td>Snort</td>
<td>ID rujukan bagi setiap tandatangan.</td>
</tr>
<tr>
<td>reference_system</td>
<td>Snort</td>
<td>Senarai rujukan sistem.</td>
</tr>
<tr>
<td>sig_class</td>
<td>Snort</td>
<td>Senarai amaran/tandatangan yang diklasifikasikan.</td>
</tr>
<tr>
<td>data</td>
<td>Snort</td>
<td>Kandungan paket.</td>
</tr>
<tr>
<td>iphdr</td>
<td>Snort</td>
<td>Maklumat IP protokol.</td>
</tr>
<tr>
<td>tcphdr</td>
<td>Snort</td>
<td>Maklumat TCP protokol.</td>
</tr>
<tr>
<td>udphdr</td>
<td>Snort</td>
<td>Maklumat UDP protokol.</td>
</tr>
<tr>
<td>icmpdr</td>
<td>Snort</td>
<td>Maklumat ICMP protokol.</td>
</tr>
<tr>
<td>opt</td>
<td>Snort</td>
<td>Pilihan IP dan TCP.</td>
</tr>
<tr>
<td>redalert</td>
<td>RedAlertIDS</td>
<td>Senarai redalert dan syarat polisi yang akan dibentuk.</td>
</tr>
<tr>
<td>redaction</td>
<td>RedAlertIDS</td>
<td>Tindakan polisi yang telah diambil oleh komponen.</td>
</tr>
</tbody>
</table>

Perhubungan antara jadual-jadual dalam komponen RedAlertIDS dan juga Snort banyak melibatkan perolehan dan perbandingan data seperti alamat IP, protokol dan juga tandatangan yang digunakan. Berikut adalah atribut yang digunakan dari komponen Snort dalam aplikasi RedAlertIDS.
Jadual 5.2 : Kegunaan pangkalan data dalam sistem RedAlert

<table>
<thead>
<tr>
<th>Jadual</th>
<th>Atribut</th>
<th>Kegunaan</th>
</tr>
</thead>
<tbody>
<tr>
<td>event.snort</td>
<td>signature timestamp</td>
<td>Digunakan oleh aplikasi untuk mendapatkan kejadian pencerobohan yang berlaku dalam rangkaian.</td>
</tr>
<tr>
<td></td>
<td>sig_sid</td>
<td>Digunakan untuk mendapatkan statik dalam konfigurasi Snort.</td>
</tr>
<tr>
<td>iphdr.snort</td>
<td>ip_src</td>
<td>ip_dst</td>
</tr>
<tr>
<td>tephdr.snort</td>
<td>tcp_sport</td>
<td>tcp_dport</td>
</tr>
<tr>
<td>udphdr.snort</td>
<td>udp_sport</td>
<td>udp_dport</td>
</tr>
</tbody>
</table>

Atribut yang dinyatakan kebanyakannya digunakan oleh RedAlertIDS bagi tujuan menghasilkan polisi yang baru. Atribut signature pada jadual event digunakan untuk mendapatkan bentuk pencerobohan yang telah berlaku dalam rangkaian.

Struktur bagi redalert dan redaction adalah seperti berikut:

**Jadual 5.3 : Struktur pangkalan data RedAlert**

<table>
<thead>
<tr>
<th>Atribut</th>
<th>Jenis Data</th>
<th>Deskripsi</th>
</tr>
</thead>
<tbody>
<tr>
<td>redalert_id</td>
<td>int (10) unsigned</td>
<td>RedAlert ID</td>
</tr>
<tr>
<td>redalert_name</td>
<td>varchar (40)</td>
<td>Nama bagi setiap bentuk pencerobohan yang hendak dikawal.</td>
</tr>
<tr>
<td>sig_sid</td>
<td>int (10) unsigned</td>
<td>Tanda tangan ID bagi kegunaan dalam sistem.</td>
</tr>
<tr>
<td>dport</td>
<td>tinyint (3) unsigned</td>
<td>Keperluan bagi elemen destinasi port untuk polisi yang akan dibuat.</td>
</tr>
<tr>
<td>sport</td>
<td>tinyint (3) unsigned</td>
<td>Keperluan bagi elemen sumber port untuk polisi yang akan dibuat.</td>
</tr>
<tr>
<td>dip</td>
<td>tinyint (3) unsigned</td>
<td>Keperluan bagi elemen destinasi IP untuk polisi yang akan dibuat.</td>
</tr>
<tr>
<td>sip</td>
<td>tinyint (3) unsigned</td>
<td>Keperluan bagi elemen sumber IP untuk polisi yang akan dibuat.</td>
</tr>
<tr>
<td>type</td>
<td>varchar(5)</td>
<td>Bentuk protokol serangan. (UDP,TCP dan ICMP)</td>
</tr>
<tr>
<td>firewall</td>
<td>varchar (40)</td>
<td>Bentuk rantaian <strong>firewall.</strong> (INPUT,OUTPUT dan FORWARD)</td>
</tr>
<tr>
<td>clearlimit</td>
<td>int (10) unsigned</td>
<td>Masa untuk polisi berada dalam rangkaian.</td>
</tr>
<tr>
<td>clearstatus</td>
<td>tinyint (3) unsigned</td>
<td>Status untuk polisi sama ada sementara atau kekal.</td>
</tr>
</tbody>
</table>

**Jadual 5.4 : Struktur pangkalan data Redaction**

<table>
<thead>
<tr>
<th>Atribut</th>
<th>Jenis Data</th>
<th>Deskripsi</th>
</tr>
</thead>
<tbody>
<tr>
<td>redaction_id</td>
<td>int (10) unsigned</td>
<td>Redaction ID.</td>
</tr>
<tr>
<td>cid</td>
<td>int (10) unsigned</td>
<td>ID bagi Kejadian.</td>
</tr>
<tr>
<td>action</td>
<td>text</td>
<td>Tindakan yang diambil. Polisi <strong>firewall.</strong></td>
</tr>
<tr>
<td>timesending</td>
<td>int (10) unsigned</td>
<td>Masa polisi dihantar.</td>
</tr>
<tr>
<td>timeclear</td>
<td>int (10) unsigned</td>
<td>Masa polisi perlu di hapuskan</td>
</tr>
<tr>
<td>clearstatus</td>
<td>tinyint (3) unsigned</td>
<td>Status polisi sama ada telah dihapuskan atau sebaliknya.</td>
</tr>
</tbody>
</table>

Bagi kedua-dua jadual ini bentuk operasi adalah berbeza. Bagi jadual redalert pentadbir perlu memasukkan nilai. Nilai bergantung kepada pentadbir untuk membuat polisi terhadap setiap bentuk pencerobohan. Ini termasuklah menentukan masa dan status polisi **firewall** dalam rangkaian. Pentadbir perlu memastikan nilai yang dimasukkan bertepatan
dengan bentuk serangan. Jika tidak polisi yang dibuat tidak berupaya menghalang penceroboh menceroboh sistem.

Redaction pula merujuk kepada setiap tindakan yang telah diambil oleh aplikasi RedAlertIDS. Setiap tindakan yang diambil akan direkodkan. Masa tindakan diambil dan masa polisi dihapuskan perlu direkodkan. Ini termasuklah status bagi polisi firewall dalam rangkaian sama ada masih berjalan atau sebaliknya.

5.4 ALGORITMA POLISI FIREWALL

Polisi firewall yang dirangka merujuk kepada jenis dan bentuk pencerobohan. Dengan demikian, proses penghasilan polisi baru bergantung kepada pangkalan data yang menyimpan syarat dan juga maklumat lain yang berkaitan.

Bentuk polisi yang dibuat bergantung kepada beberapa syarat dan algoritma dalam penghasilannya. Berikut merupakan algoritma tersebut:

MULA redalert_polisi()
GET protocol_type
IF protocol_type==TCP THEN
   GET tcp hdr
   ELSE IF protocol_type==UDP THEN
      GET udp hdr
   END IF
END IF
CREATE new_polisi
STORE new_polisi
TAMAT redalert_polisi()

Penghasilan polisi bergantung kepada protokol yang digunakan. Jika ianya menggunakan TCP maka polisi akan membentuk firewall yang menghalang protokol TCP. Sebaliknya
jika protokol UDP dan ICMP digunakan maka polisi firewall akan menghalangi protokol ICMP dan UDP.

Penghasilan polisi baru bergantung kepada atribut yang disimpan dalam pangkalan data termasuk dport, sport, dip, sip, type dan juga firewall. Contoh polisi yang dibentuk adalah seperti berikut:

i.  
/bin/iptables -A INPUT -p UDP -s 202.5.6.2 --sport 123 -j DROP

Ini bermakna setiap komunikasi kepada rantaian INPUT yang menggunakan protokol UDP dari sumber IP 202.5.6.2 melalui port 123 akan dihalangi.

ii.  
/bin/iptables -A OUTPUT -p TCP -d 202.5.6.2 --dport 123 -j DROP

Ini bermakna setiap komunikasi kepada rantaian OUTPUT yang menggunakan protokol TCP ke destinasi IP 202.5.6.2 melalui port 123 akan dihalangi.

Setiap serangan dan juga pencerobohan yang berlaku akan memberi maklumat seperti IP dan juga port yang digunakan. Berdasarkan IP dan port, setiap bentuk pencerobohan berkaitan boleh dihalangi.

Selepas polisi dihantar, polisi dan masa penghantaran disimpan untuk tujuan semakan dan tujuan pengemaskinian masa depan. Setiap polisi yang dibentuk dan dihantar akan
mengalami dua proses iaitu proses tambahan dan pembuangan. Setiap polisi akan dihantar sebanyak dua kali bergantung jika ianya perlu melalui proses pembuangan. Masa polisi berada dalam rangkaian tertakluk kepada syarat yang ditentukan oleh pentadbir. Isyarat polisi kedua yang dihantar adalah sama seperti isyarat polisi pertama dengan perbezaan pada simbol –A dan –D.

Polisi yang dihantar kali pertama:

/sbin/iptables -A OUTPUT -p TCP -d 202.5.6.2 --dport 123 -j DROP

Polisi yang dihantar kali kedua:

/sbin/iptables -D OUTPUT -p TCP -d 202.5.6.2 --dport 123 -j DROP

Perbezaan masa antara kedua-dua penghantaran bergantung kepada pentadbir dan dikira dalam unit saat. Unit masa yang disimpan dalam pangkalan data ditukar dalam unit saat dalam kiraan sehari. Ini bermakna nilai tidak sama dengan nilai saat, minit dan jam dalam kiraan biasa. Ianya bertujuan untuk memberi satu pendekatan pengiraan yang tepat dan kurang ralat. Pengiraan masa menggunakan fungsi time.h dalam aturcara C. Fungsi strftime digunakan untuk mengubah format tarikh dan masa dalam bentuk string.

Pengiraan masa penghantaran berdasarkan persamaan berikut:

\[
\begin{align*}
\text{String (Y)} & = \text{Tahun & Hari dalam tahun berkenaan} \\
\text{String (X)} & = \text{y & String(\text{Int } ((\text{Jam} \times 3600) + (\text{Minit} \times 60) + \text{saat}))} \\
\text{Masa hantar} & = \text{Unsigned Long Int (X)}
\end{align*}
\]
Ianya ditulis bertujuan untuk mengelakkan ralat pengiraan masa yang sering terjadi seperti pergerakan hari dan bulan dalam setahun. Ianya dapat dibezakan dengan lebih tepat berbanding pengiraan masa sahaja.

**Jadual 5.5 : Pengiraan masa menggunakan timered()**

<table>
<thead>
<tr>
<th>Masa Nyata</th>
<th>Pengiraan biasa</th>
<th>Pengiraan Timered()</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. 31 Disember 2002 23:59:01</td>
<td>86341</td>
<td>0234486341</td>
</tr>
<tr>
<td>2. 1 Januari 2003 00:00:01</td>
<td>00001</td>
<td>0300100001</td>
</tr>
<tr>
<td>3. 1 Januari 2003 05:00:01</td>
<td>18001</td>
<td>0300118001</td>
</tr>
<tr>
<td>4. 2 Januari 2003 14:59:01</td>
<td>53941</td>
<td>0300253941</td>
</tr>
<tr>
<td>5. 2 Januari 2003 15:00:00</td>
<td>54000</td>
<td>0300254000</td>
</tr>
<tr>
<td>6. 3 Januari 2003 14:59:01</td>
<td>53941</td>
<td>0300353940</td>
</tr>
</tbody>
</table>


Dari perbandingan di atas, jelas menunjukkan pengiraan timered() lebih baik berbanding pengiraan biasa. Pengiraan tepat tanpa ralat dibuktikan dengan kenyataan perbandingan berikut:

dapat dibezakan walaupun ianya ditambah 120 saat atau 86400 saat sekalipun.


Perbezaan dapat dilakukan dengan tepat dan jitu kerana masa yang tukar bergantung kepada tahun, kedudukan hari dalam tahun berkenaan dan masa dalam saat. Tiga elemen ini dapat membezakan secara total bagi proses pengemaskinian polisi dalam jangka masa yang panjang.

5.5 FUNGSIAN TRANSMISI DATA SISTEM REDALERT

MULA redalert_signal()
  RECEIVED data
  SET config
  ENCODE data
  SEND data
  Redalert_unicast(data)
TAMAT redalert_signal()


MULA redalert_unicast()
  RECEIVED data
  SET config
  WHILE not EMPTY(unicast host)
    SEND DATA to unicast host
  END WHILE
TAMAT redalert_unicast()

Proses penghantaran kepada hos unicast dilakukan dengan membaca bilangan hos unicast. Proses penghantaran ini berulang sehingga hos unicast dalam konfigurasi fail habis. Tertakluk kepada konfigurasi, hos unicast boleh terdiri daripada 1 atau lebih.

Aplikasi RedAlertIDS adalah agen bagi fungsi mendapatkan data penceroboh, membentuk polisi dan menghantarn polisi. Aplikasi tidak membuat tindakan susulan dengan menghalang penceroboh, sebaliknya memberi maklumat kepada agen penerima.
untuk menjalankan tindakan selanjutnya. Tindakan selanjutnya akan melibatkan 2 aplikasi yang berbeza yang diletakkan di dua lokasi yang berlainan.

i. RedAlertMulti

Merupakan agen yang akan menerima transmisi multicast yang dihantar oleh RedAlertIDS. Agen ini menerima dan menterjemahkan semula data yang dihantar dengan menggunakan algoritma XOR yang sama. Proses penterjemahan adalah mudah dan ringkas. Data yang diterjemah dijadikan polisi firewall yang baru. Lokasi ditempatkan disemua hos yang berkenaan dengan aplikasi RedAlertIDS.

```
MULA redalertmulti()
  SET config
  LISTENING
  WHILE not EMPTY(data)
    RECEIVED data
    DECODE data
    RUN SYSTEM(data)
  END WHILE
TAMAT redalertmulti()
```

ii. RedAlertUni

MULA redalertuni()
SET config
LISTENING
WHILE not EMTPY(data)
   RECEIVED data
   Redalertmulti(data)
END WHILE
TAMAT redalertuni()

5.6 PENDEKATAN SISTEM REDALERT DALAM RANGKAIAN

Aplikasi RedAlertIDS merupakan aplikasi integrasi beberapa aplikasi yang sedia ada dengan aplikasi baru yang dicipta. Penerapan aplikasi sedia ada membantu menghasilkan satu mekanisme yang baru dalam menghalang penceroboh dari menceroboh. Bentuk pencerobohan yang dihalang merupakan pencerobohan yang dianggap sebagai serangan yang boleh menyebabkan kegagalan rangkaian. Pentadbir berupaya menentukan bentuk serangan yang harus dihalang.

Gabungan Snort, Multicast, Unicast, iptables Firewall dalam sistem operasi Linux memberi satu platform dalam mengawal serangan. Pangkalan data MySQL yang digunakan merupakan pangkalan data asal yang juga digunakan oleh aplikasi Snort.

Beberapa pendekatan boleh diambil dalam merealisasikan sistem ini. Dua bentuk rekabentuk rangkaian yang dicadangkan iaitu melibatkan satu LAN dan juga beberapa LAN dalam VLAN. Sistem ini juga berupaya bertindak dalam bentuk yang lebih besar dengan pendekatan apa jua topologi rangkaian termasuk penggunaan rangkaian besar seperti penggunaan WAN.

WGAZ01006 120
Elemen yang memainkan peranan penting dalam rekabentuk rangkaian ialah agen multicast dan unicast. Agen unicast yang digunakan bertujuan sebagai laluan kepada transmisi multicast untuk ke subnet yang lain. Transmisi multicast di konfigurasi supaya ianya cuma dapat menghantar data dalam subnet yang sama sahaja bagi mengelak sebarang masalah transmisi ke luar rangkaian. Oleh yang demikian agen unicast yang digunakan bersamaan dengan n-1, dengan n merujuk kepada bilangan rangkaian subnet.

Dua pendekatan yang berbeza dalam implementasi sistem RedAlert iaitu menggunakan 1 LAN atau beberapa LAN. Penggunaan ini dinamakan Uni RedAlert mewakili 1 LAN dan Multi RedAlert bagi merujuk beberapa LAN. Perbezaan dalam implementasi sistem RedAlert adalah seperti berikut:

<table>
<thead>
<tr>
<th>Jadual 5.6 : Perbezaan fungsi Uni RedAlert dan Multi RedAlert</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LAN</strong></td>
</tr>
<tr>
<td>Agen Unicast</td>
</tr>
<tr>
<td>Agen Multicast</td>
</tr>
<tr>
<td>Snort IDS</td>
</tr>
<tr>
<td>Firewall</td>
</tr>
<tr>
<td>Bentuk transmisi rangkaian</td>
</tr>
<tr>
<td>Masa pengemaskian.</td>
</tr>
<tr>
<td>Keberkesanan</td>
</tr>
</tbody>
</table>

Secara keseluruhan jika bilangan subnet yang hendak digunakan melebihi daripada 1 subnet maka dua bentuk transmisi digunakan iaitu secara Multicast dan Unicast. Penggunaan transmisi unicast digunakan untuk memberi laluan transmisi multicast yang
Mekanisme Pengemaskinian Polisi Keselamatan *Firewall* dengan Pendekatan Sistem Pengesan Pencerobohan dan Protokol Multicast
dihalang dari keluar dari subnet. Ianya digunakan sebagai terowong yang menghubungkan 2 atau rangkaian subnet.

Algorithm *RedAlert* memberi satu mekanisme baru untuk proses pengemaskian polisi *firewall* secara transmisi multicast dengan kehadiran IDS.