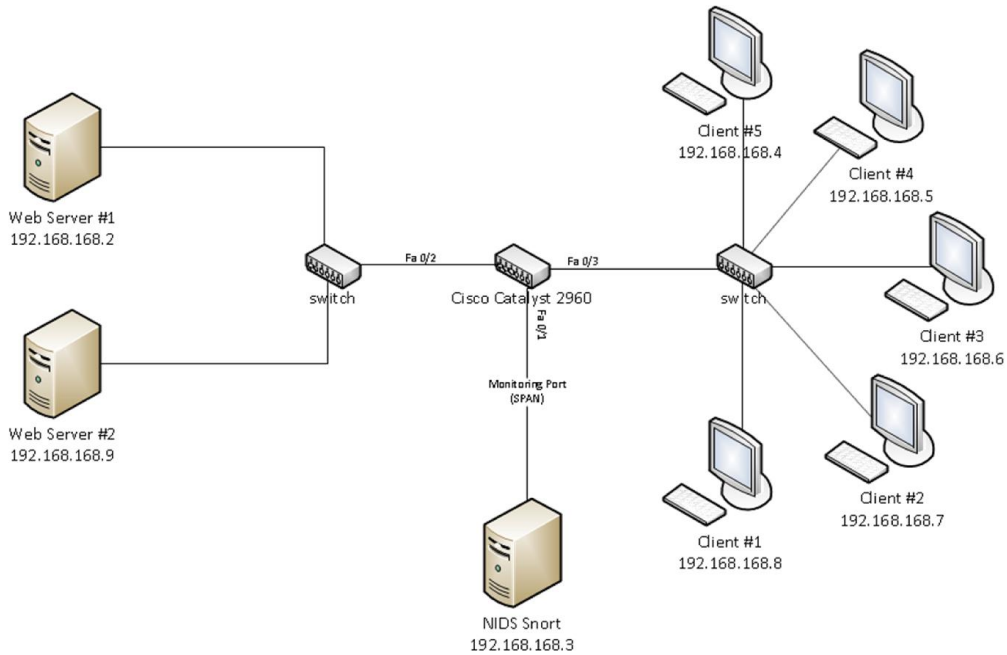
BAB 4
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi

Tahap implementasi merupakan penerapan dari tahap analisis dan perancangan. Pada tahap ini perancangan-perancangan sistem pada penelitian yang telah dibuat akan diimplementasikan dalam bentuk konfigurasi dan pengkodean. Dengan demikian dapat diketahui apakah perancangan tersebut dapat berjalan dengan baik atau tidak.

4.1.1 Implementasi Arsitektur Jaringan Pengujian

Implementasi ini merupakan penerapan dari perancangan arsitektur jaringan pada **Error! Reference source not found.**, pada perancangan tersebut digambarkan bagaimana setiap komputer dan perangkat lainnya terhubung. Berikut ini adalah Gambar 4.1 arsitektur pengujian yang akan diimplementasikan yang telah diberikan pengalamatan untuk setiap komputernya.



Gambar 4.1 Arsitektur Jaringan Pengujian dengan *IP address*

4.1.1.1 Implementasi Perangkat Keras

Implementasi ini merupakan penjelasan mengenai spesifikasi perangkat keras yang digunakan pada arsitektur jaringan yang akan diimplementasikan. Pada gambar aritektur pengujian (Gambar 4.1) terdapat beberapa perangkat keras yaitu *switch unmanagable*, *switch manageable*, komputer *client*, komputer *server*, komputer *server* IDS dan kabel UTP. Setiap perangkat keras tersebut memiliki spesifikasi yang dijelaskan pada Tabel 4.1 sampai dengan Tabel 4.4.

Tabel 4.1 Spesifikasi *Switch Managable* Cisco 2960

No	Spesifikasi	Deskripsi
1	Ports	24 Port
2	Forwarding bandwidth	16 Gbps
3	Switching bandwidth	32Gbps
4	Flash memory	32 MB
5	Memory DRAM	64 MB
6	Max VLANs	255
7	VLAN IDs	4000
8	Maximum transmission unit (MTU)	Up to 9000 bytes
9	Jumbo frames	9018 bytes (2960G only)
10	Dimension	1.73 x 17.7 x 9.52

Tabel 4.2 Spesifikasi *Switch Unmanagable*

No	Spesifikasi	Deskripsi
1	Type	Dlink DES-1008A
2	Ports	8x 10/100Mbps RJ45 ports (Auto MDI/MDIX)

No	Spesifikasi	Deskripsi
3	Backbound Bandwidth	1.6 Gbps
4	Dimension	128(W) x 68.5(H) x 25.4(D)
5	Forwarding Rate	10 BASE-T (14880pps/ port) , 100 BASE-TX (148800pps/ port)
6	IEEE802.3x Flow Control	Full-Duplex Mode (20/200 Mpbs) and backpressure for Half-Duplex Mode (10/100 Mpbs)

Tabel 4.3 Spesifikasi Perangkat Keras Komputer *Client* 1 s.d. *Client* 5

No	Spesifikasi	Client 1,2	Client 3,4,5
1	Processor	Core 2 Duo	Core 2 Duo
2	RAM	2 Gb DDR2	2 Gb DDR3
3	VGA	GeForce 8500GT	ATI Radeon HD4350
4	Harddisk	320 Gb	320 Gb
5	NIC	10/100Mb/s	10/100Mb/s

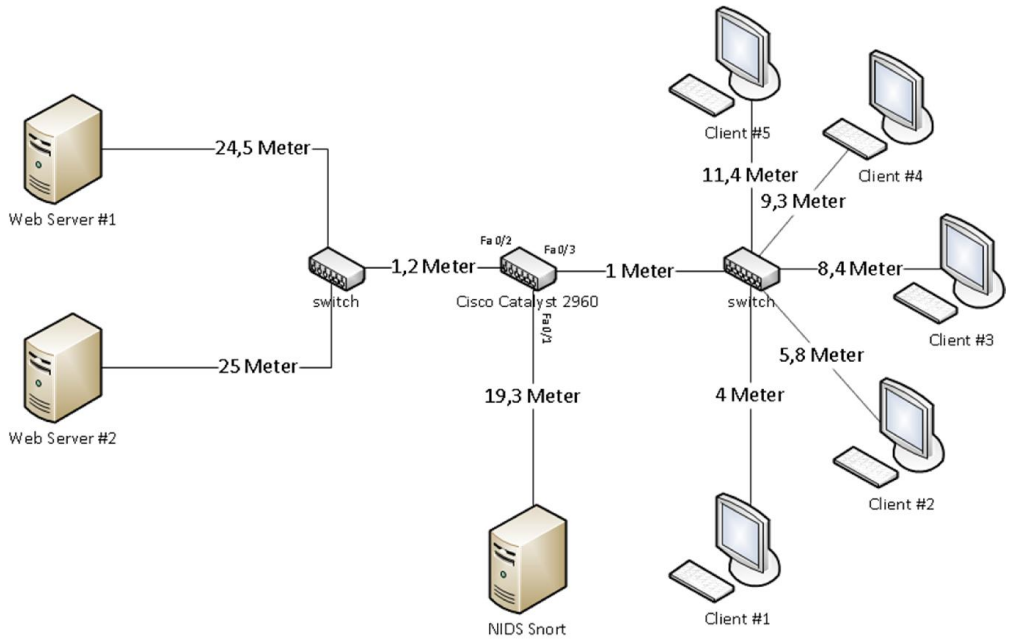
Tabel 4.4 Spesifikasi Perangkat Keras *Web Server* 1 dan *Web Server* 2

No	Spesifikasi	Web Server 1	Web Server 2
1	Processor	Core i3	Core i5
2	RAM	6 Gb DDR3	4Gb DDR3
3	VGA	Intel 1024 Mb	Geforce 9500GT
4	Harddisk	500 Gb	1 Tb
5	NIC	10/100Mb/s	10/100Mb/s

Tabel 4.5 Spesifikasi Perangkat Keras Komputer IDS Snort

No	Spesifikasi	Deskripsi
1	Processor	Core 2 Duo
2	RAM	2 Gb DDR3
3	VGA	ATI Radeon HD4350
4	Harddisk	320 Gb
5	NIC	10/100Mb/s

Untuk spesifikasi kabel UTP yang digunakan yaitu Belden Cat5e, kabel UTP ini menghubungkan setiap perangkat-perangkat jaringan dengan panjang kabel dari setiap *node* ke *node* yang digambarkan pada Gambar 4.2.



Gambar 4.2 Panjang Kabel Pada Arsitektur Jaringan

4.1.1.2 Implementasi Perangkat Lunak

Implementasi ini merupakan penjelasan dari aplikasi dan sistem operasi yang digunakan pada setiap perangkat keras dari arsitektur jaringan. Berikut ini

adalah aplikasi dan sistem operasi yang digunakan pada setiap perangkat, yang dipaparkan pada Tabel 4.6 sampai dengan Tabel 4.8.

Tabel 4.6 Perangkat Lunak Pada *Web Server* 1 dan *Web Server* 2

No	Spesifikasi	Web Server 1	Web Server 2
1	Sistem Operasi	Linux Ubuntu 12.04 32 bit	Linux Mint
2	Database	MySQL 5.5.37	MySQL 5.5.37
3	Web Server	Apache/2.2.22	Apache/2.2.22

Tabel 4.7 Perangkat Lunak Pada Komputer IDS

No	Spesifikasi	Deskripsi
1	Sistem Operasi	Linux Ubuntu 14.04 32 bit
2	Database	MySQL 5.5.37
3	Web Server	Apache/2.4.7
4	Aplikasi IDS	Snort 2.9.6

Tabel 4.8 Perangkat Lunak Pada Komputer *Client* 1 s.d. *Client* 5

No	Spesifikasi	Client 1,3,4,5	Client 2
1	Sistem Operasi	Windows 7 Professional	Linux Mint
2	Browser	Mozilla Firefox 30.0	Mozilla Firefox 30.0
3	Tools	Zenmap 6.40	Terminal

4.1.1.3 Implementasi dan Konfigurasi Arsitektur Jaringan

Implementasi ini merupakan penjelasan bagaimana cara mengkonfigurasi cisco *router* dan pengalamatan pada masing-masing perangkat komputer.

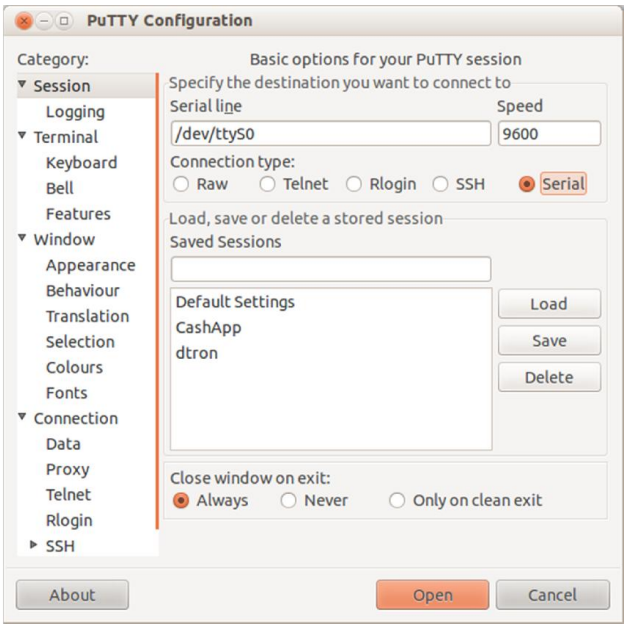
1. Konfigurasi Cisco Catalyst 2960

   Konfigurasi cisco menggunakan *socket console* dan aplikasi putty, konfigurasinya hanya mengkonfigurasi *port* fa0/1, fa0/2, dan fa0/3. *Port* fa0/1 dikofigurasi sebagai SPAN (*Switch Port Analyzer*) atau *port mirorring* dari *port* fa0/2 dan fa0/3, sehingga *traffic* data yang melalui jalur tersebut akan dikirimkan juga pada *port mirroring* tersebut (port fa0/1).

   Berikut ini langkah-langkah konfigurasi SPAN pada cisco catalyst 2960 :

   a. *Setting* Putty

      Pilih tab connection type serial, serial line /dev/ttyS0 dan speed 9600, selanjutnya tekan tombol open.



Gambar 4.3 Konfigurasi Putty

   b. Konfigurasi SPAN (*Switched Port Analyzer*) Cisco

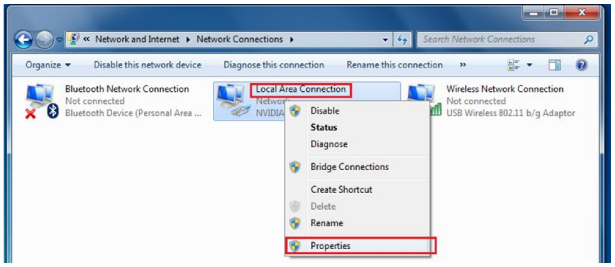
```
Router1(config)#monitor session 1 source
interface fa0/2, fa0/3 both
Router1(config)#monitor session 1 destination
interface fa0/1
```

2. Konfigurasi *IP address* Perangkat Keras

   Konfigurasi ini adalah cara untuk mengkonfigurasi *ip address* pada setiap perangkat komputer dalam arsitektur jaringan, pada dasarnya cara konfigurasi *ip address* setiap komputer sama jika sistem operasi yang digunakan sama. Pada penelitian ini menggunakan 2 jenis sistem operasi yaitu windows 7 dan linux. Berikut ini cara mengkonfigurasi *ip address* pada windows 7 (*client* 1,3,4,5) dan linux (*client* 2, IDS *server*, web *server* 1,2).

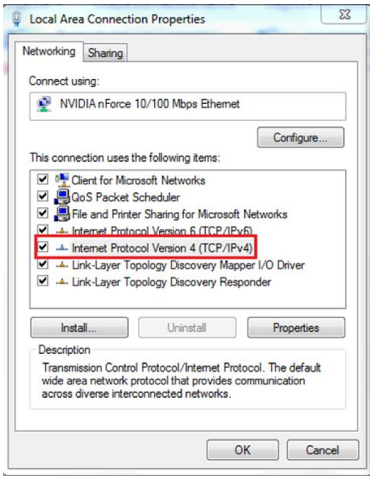
   1. Konfigurasi *IP address* pada windows 7.

      a. Masuk ke menu control panel → network and internet → network and sharing center → change adapter setting → pada local area connection klik kanan pilih properties.



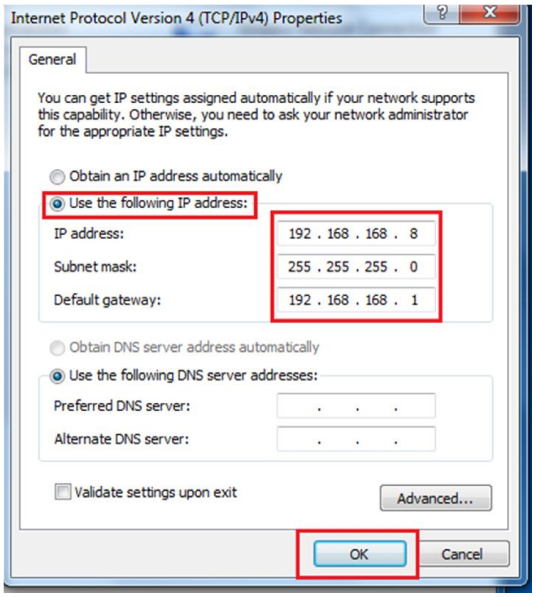
Gambar 4.4 Konfigurasi IP pada windows 7

      b. Setelah muncul dialog local area connection properties lalu pilih internet protocol version 4 (TCP/Ipv4) → Properties.



Gambar 4.5 Kotak Dialog Local Area Connection Properties

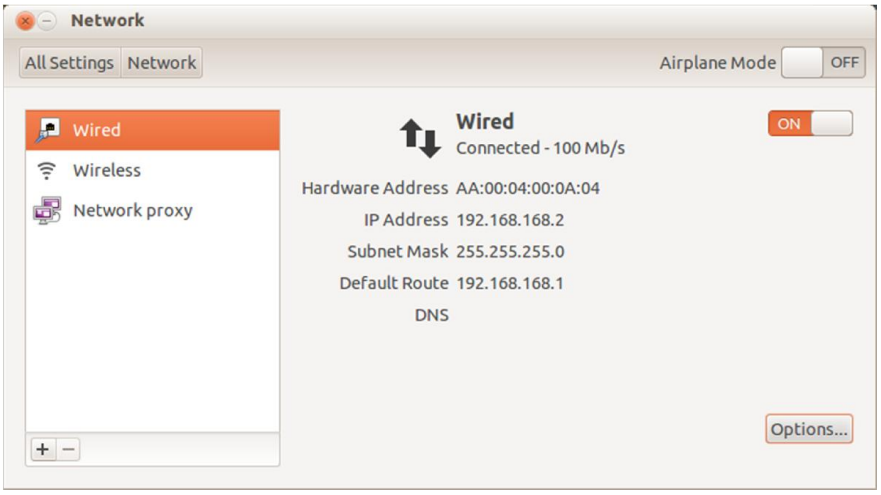
c. Pada kolom isian *ip address, subnet mask*, *default gateway* diisikan sesuai dengan pada Gambar 4.1. Dibawah ini merupakan gambar dari konfigurasi *ip address* pada *client* 5.



Gambar 4.6 Konfigurasi IP *Client* 5

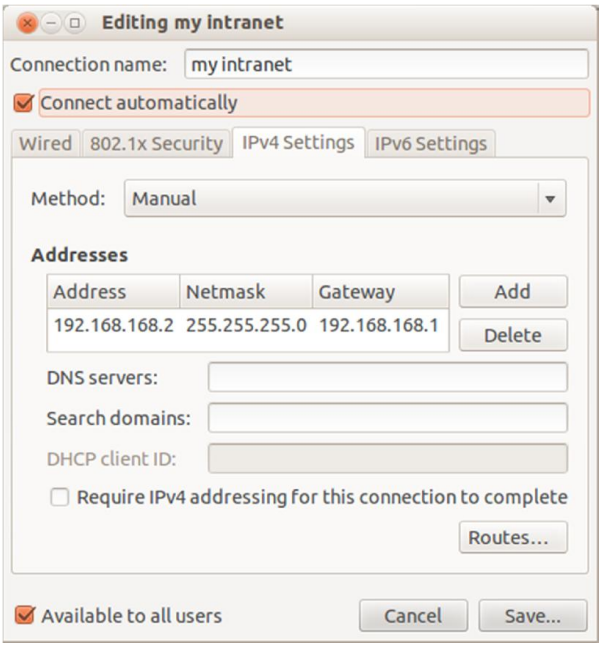
2. Konfigurasi *IP address* pada linux ubuntu (IDS *Server*, *Web Server* 1,2).

a. Masuk ke menu system setting → network → options.



Gambar 4.7 Dialog Network Linux Ubuntu

b. Pada kolom isian *address*, *netmask* dan *gateway* masukkan isian sesuai dengan *ip addess* pada arsitektur jaringan, lalu tekan tombol save. Gambar 4.8 adalah konfigurasi *IP address* untuk *web server* 1.



Gambar 4.8 Konfigurasi IP Web Server 1

4.1.2 Implementasi Aplikasi

Implementasi ini merupakan penjelasan mengenai cara menginstalasi aplikasi IDS snort, pengimplementasian perancangan dari *rule* snort, dan pengimplementasian perancangan aplikasi *monitoring* IDS snortbow.

4.1.2.1 Implementasi Snort

Implementasi ini dilakukan pada komputer *server* IDS pada sistem operasi linux ubuntu 14.04, berikut ini adalah tahapan-tahapan instalasi aplikasi snort yang digunakan sebagai aplikasi IDS.

1. *Download package* LAMP (Apache, MySQL, PHP), digunakan untuk menjalankan aplikasi snortbow. Pada *terminal* (*user root*) ketikkan perintah :

```
Apt-get -y install apache2 libapache2-mod-php5
mysql-server mysql-common mysql-client php5-mysql
libmysqlclient-dev php5-gd php-pear libphp-adodb
php5-cli
```

Ikuti perintah selanjutnya sesuai dengan instruksi pada aplikasi yang diinstall.

2. *Install library* pendukung snort. Pada *terminal* (*user root*) ketikkan perintah:

```
apt-get -y install libwww-perl libnet1 libnet1-dev
libpcre3 libpcre3-dev autoconf libcrypt-ssleay-perl
libtool libssl-dev build-essential automake gcc
make flex bison
```

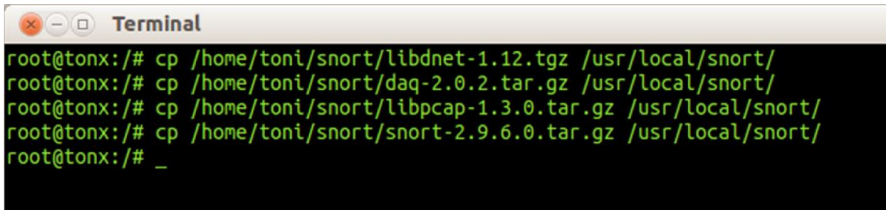
Ikuti perintah selanjutnya sesuai dengan instruksi pada aplikasi yang di-*install*.

3. *Download* aplikasi pendukung snort dan snort 2.9.6. Untuk menjalankan snort ada beberapa *library* yang perlu di-*download*, *library* tersebut antara lain :
   a. Libdnet yang dapat di-*download* di http://libdnet.googlecode.com/files/libdnet-1.12.tgz.
   b. Daq-2.0.2 yang dapat di-*download* di http://sourceforge.net/projects/snort/files/snort/daq-2.0.2.tar.gz/download.
   c. Libpcap 1.3.0 yang dapat di-*download* di http://www.tcpdump.org/release/libpcap-1.3.0.tar.gz.
   d. Snort 2.9.6 yang dapat di-*download* di https://www.snort.org/downloads/snort/snort-2.9.6.1.tar.gz.
4. Instalasi snort

   Dari hasil *download* aplikasi pendukung snort dan snort 2.9.6 tersebut di-*copy* ke *folder* usr/local/snort. Pada *terminal* ketikkan perintah :

```
# cp /home/toni/snort/libdnet-1.12.tgz /usr/local/snort/
# cp /home/toni/snort/daq-2.0.2.tar.gz /usr/local/snort/
# cp /home/toni/snort/libpcap-1.3.0.tar.gz
  /usr/local/snort/
```

```
# cp /home/toni/snort/snort-2.9.6.0.tar.gz
  /usr/local/snort/
```



Gambar 4.9 *Copy file* Aplikasi Pendukung Snort

a. *Install* libdnet 1.12

```
# cd usr/local/snort
# tar zxvf libdnet-1.12.tgz
# cd libdnet-1.12
# ./configure
# make
# make install
# ln -s /usr/local/lib/libdnet.1.0.1
  /usr/lib/libdnet.1
```

b. *Install* daq 2.02

```
# cd usr/local/snort
# tar zxvf daq-2.0.2.tar.gz
# cd daq-2.0.2
# ./configure
# make
# make install
```

c. *Install* libcap 1.3.0

```
# cd usr/local/snort
```

```
# tar zxvf libpcap-1.3.0.tar.gz
# cd libpcap-1.3.0
# ./configure
# make
# make install
# echo "/usr/local/lib" >> /etc/ld.so.conf
# ldconfig -v
```

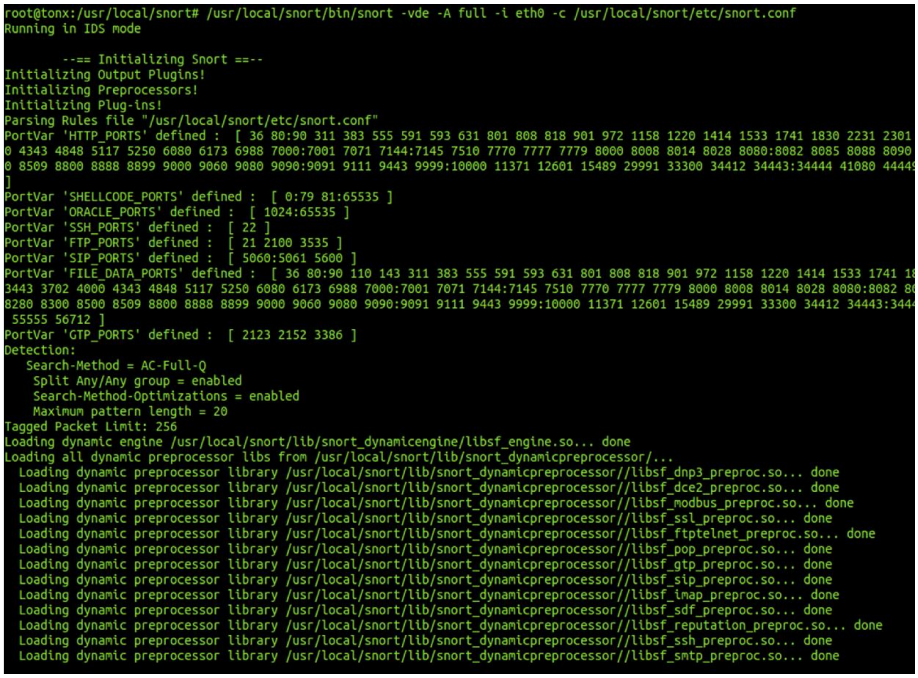
d. *Install* snort 2.9.6

```
# cd usr/local/snort
# tar zxf snort-2.9.6.0.tar.gz
# cd snort-2.9.6.0
# ./configure –prefix=/usr/local/snort –enable-
  sourcefire
# make
# make install
# mkdir /var/log/snort
# mkdir /var/snort
# groupadd snort
# useradd -g snort snort
# chown snort:snort /var/log/snort
```

5. Menjalankan snort

Aplikasi snort dijalankan di dalam *terminal*, untuk menjalankan aplikasi snort ketikkan perintah :

```
# /usr/local/snort/bin/snort -vde -A full -i eth0
  –c /usr/local/snort/etc/snort.conf
```



Gambar 4.10 Menjalankan Aplikasi Snort

4.1.2.2 Implementasi *Rules* Snort

Implementasi ini merupakan pengimplementasian dari perancangan pada poin **Error! Reference source not found.**. *Rule* snort digunakan untuk mendeteksi adanya serangan atau aktivitas yang dilakukan oleh *client-client* terhadap *web server*. Berikut ini adalah *rule-rule* yang digunakan untuk mendeteksi aktivitas-aktivitas tersebut.

1. *Rule* Deteksi SQL *injection*

```
alert tcp any any -> any any (msg:"Serangan SQL injection : Memasukan ' pada URL"; content:"'";http_uri; sid:1000011;)

alert tcp any any -> any any (msg:"Serangan SQL injection : Memasukan kata "union" pada URL"; content:"union";http_uri; nocase; sid:1000012;)

alert tcp any any -> any any (msg:"Serangan SQL injection : Memasukan kata "database()" pada URL"; content:"database()";http_uri; nocase; sid:1000013;)

alert tcp any any -> any any (msg:"Serangan SQL
```

```
injection : Ada yang akses database information_schema"; content:"information_schema"; http_uri; nocase; sid:1000014;)

alert tcp any any -> any any (msg:"Serangan SQL injection : Mencari jumlah kolom"; content:"order by";http_uri; nocase; sid:1000015;)

alert tcp any any -> any any (msg:"Ada Yang mengakses Login Page"; content:"admin"; nocase; sid:1000016;)

alert tcp any any -> any any (msg:"Ada Yang Masuk ke Backend"; content:"home.php"; nocase; sid:1000017;)

alert tcp any any -> any any (msg:"Serangan SQL injection : Memasukkan Character SQL"; flow:to_server,established;pcre:"/(\%27)|(\')|(\-\-)|(%23)|(#)/i"; classtype:Web-application-attack; sid:1000018; rev:5;)
```

2. *Rule* Deteksi *Cross Site Scripting*

```
alert tcp any any -> any any (msg:"Cross-site scripting attempt"; flow:to_server,established; pcre:"/((\%3C)|<)((\%2F)|\/)*[a-z0-9\%]+((\%3E)|>)/i"; classtype:Web-application-attack; sid:1000100; rev:5;)
```

3. *Rule* Deteksi *Port Scanning*

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN FIN"; flow:stateless; flags:F,12; reference:arachnids,27; classtype:attempted-recon; sid:621; rev:7;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN NULL"; flow:stateless; ack:0; flags:0; seq:0; reference:arachnids,4; classtype:attempted-recon; sid:623; rev:6;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN SYN FIN"; flow:stateless; flags:SF,12; reference:arachnids,198; classtype:attempted-recon; sid:624; rev:7;)
```

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any
(msg:"SCAN XMAS"; flow:stateless; flags:SRAFPU,12;
reference:arachnids,144; classtype:attempted-recon;
sid:625; rev:7;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any
(msg:"SCAN TONX nmap XMAS"; flow:stateless;
flags:FPU,12; reference:arachnids,30;
classtype:attempted-recon; sid:1228; rev:7;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any
(msg:"SCAN synscan portscan"; flow:stateless;
flags:S,12; id:39426; reference:arachnids,441;
classtype:attempted-recon; sid:630; rev:7;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 22
(msg:"SCAN SSH Version map attempt";
flow:to_server,established;
content:"Version_Mapper"; nocase;
classtype:network-scan; sid:1638; rev:5;)

alert udp $EXTERNAL_NET any -> $HOME_NET 1900
(msg:"SCAN UPnP service discover attempt";
content:"M-SEARCH "; depth:9;
content:"ssdp|3A|discover"; classtype:network-scan;
sid:1917; rev:6;)
```

4. *Rule* Deteksi DDOS (*SYN Flooding, Ping Large Packet*)

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any
(msg:"ICMP Large ICMP Packet"; dsize:>800;
reference:arachnids,246; classtype:bad-unknown;
sid:499; rev:4;)

alert tcp any any -> any any (flags: S; msg:"DDOS
SYN Flooding"; flow: stateless; threshold: type
both, track by_src, count 70, seconds 10;
sid:10001;rev:1;)
```

4.1.2.3 Implementasi Basis Data Snortbow

Implementasi ini merupakan pengimplementasian *database* yang digunakan oleh aplikasi snortbow. Pada impelementasi ini akan dijelaskan

beberapa sintax MySQL yang digunakan untuk membuat table-table dari *database* snortbow. Berikut ini adalah sintax-sintax MySQL tersebut.

1. Membuat *table* alert

```
CREATE TABLE IF NOT EXISTS `alert` (
  `datetime_start` datetime NOT NULL,
  `datetime` datetime NOT NULL,
  `id` varchar(20) NOT NULL,
  `message` varchar(255) NOT NULL,
  `mac_src` varchar(17) NOT NULL,
  `mac_dest` varchar(17) NOT NULL,
  `ip_src` varchar(20) NOT NULL,
  `ip_dest` varchar(20) NOT NULL,
  `port_src` int(11) NOT NULL,
  `port_dest` int(11) NOT NULL,
  `protocol` varchar(10) NOT NULL,
  `alert_full` text NOT NULL,
  `exec` bigint(20) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

2. Membuat *table* email

```
CREATE TABLE IF NOT EXISTS `email` (
  `e_name` varchar(30) NOT NULL,
 `e_mail` varchar(50) NOT NULL,
  PRIMARY KEY (`e_mail`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

3. Membuat *table* information

```
CREATE TABLE IF NOT EXISTS `information` (
  `snort_active` varchar(1) NOT NULL,
  `last_update` datetime NOT NULL,
  `last_attack` datetime NOT NULL,
```

```
  `last_email` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

4. Membuat *table* rule_detail

```
CREATE TABLE IF NOT EXISTS `rules_detail` (
  `id` int(11) NOT NULL,
  `rule` text NOT NULL,
  `file_rule` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

5. Membuat *table* rule_file

```
CREATE TABLE IF NOT EXISTS `rules_file` (
  `id_file_rule` int(11) NOT NULL,
  `file_rule` varchar(50) NOT NULL,
  PRIMARY KEY (`id_file_rule`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

6. Membuat *table* user

```
CREATE TABLE IF NOT EXISTS `user` (
  `user` varchar(30) NOT NULL,
  `password` varchar(50) NOT NULL,
  `last_login` datetime NOT NULL,
   PRIMARY KEY (`user`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

7. Membuat *table* sum_of_attack

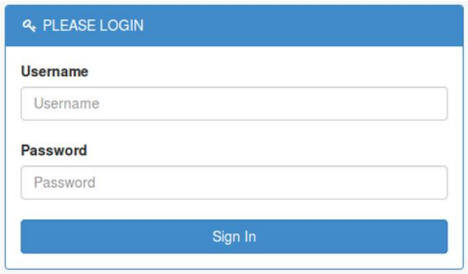
```
CREATE TABLE IF NOT EXISTS `sum_of_attack` (
  `datetime` datetime NOT NULL,
  `attack` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

4.1.2.4 Implementasi Antar Muka Snortbow

Implementasi ini merupakan implementasi dari perancangan antar muka pada poin **Error! Reference source not found.**. Dari perancangan tersebut telah diterapkan ke dalam pengkodean berbasis *web*. Berikut ini adalah antra muka dari aplikasi snortbow.

1. Halaman Login

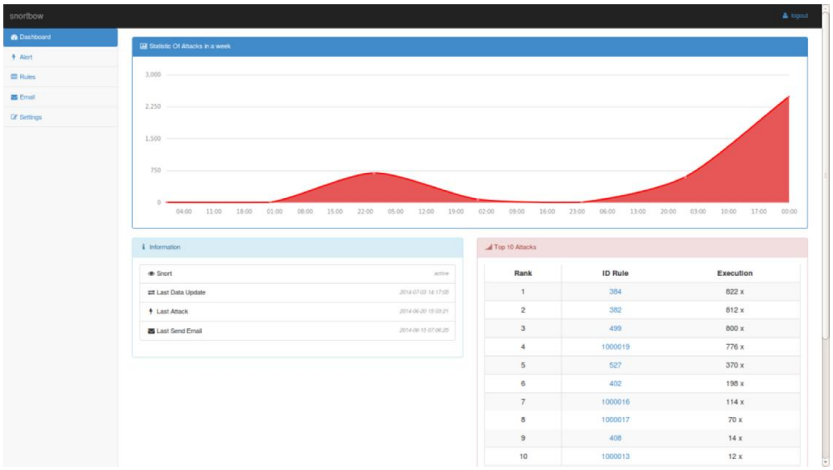
   Halaman ini digunakan untuk melakukan otentifikasi *username* dan *password* sebelum masuk ke dalam aplikasi.



Gambar 4.11 Antarmuka Halaman

2. Halaman Dashboard

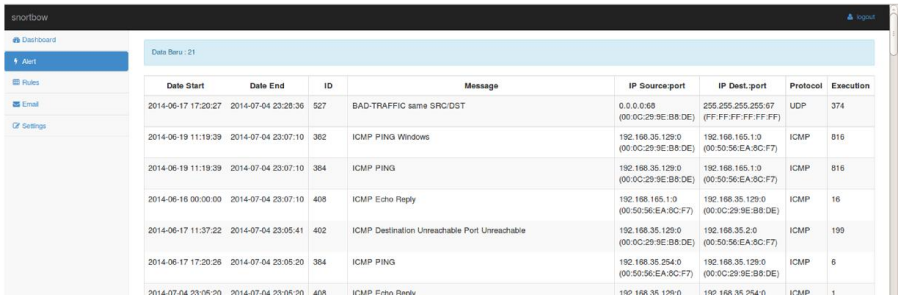
   Halaman ini digunakan untuk melihat grafik dan kesimpulan informasi-informasi.



Gambar 4.12 Antarmuka Halaman Dashboard

3. Halaman Alert

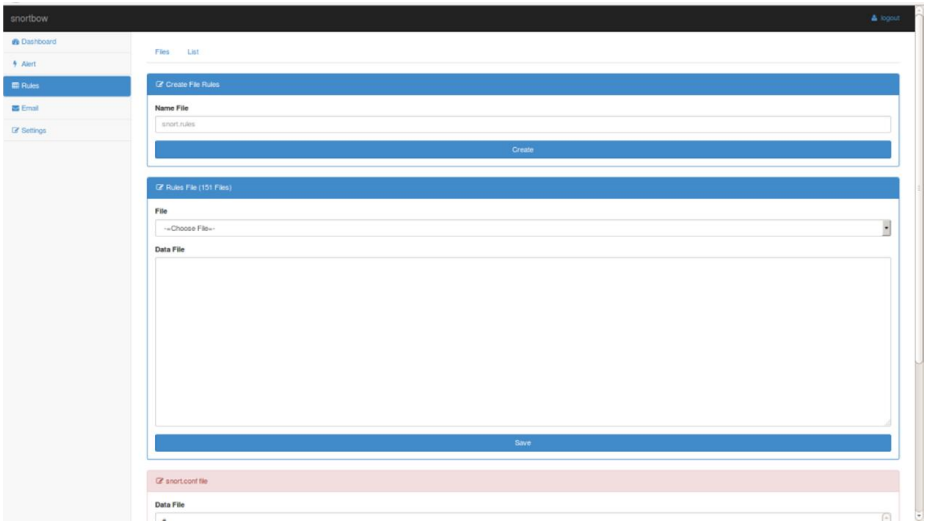
   Halaman ini digunakan untuk melihat serangan-serangan yang telah di-*log* oleh snort.



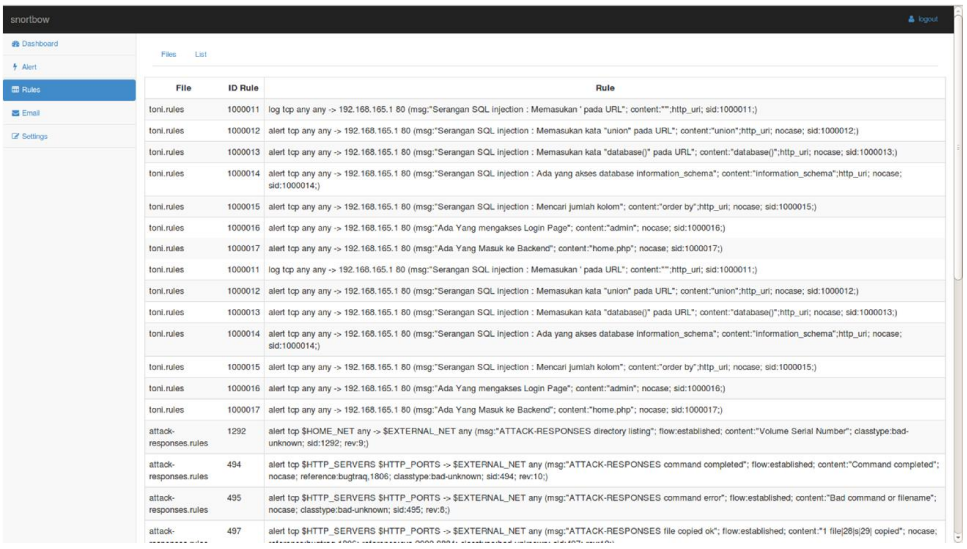
Gambar 4.13 Antarmuka Halaman Alert

4. Halaman Rules

   Halaman ini digunakan untuk menambah dan mengubah *file rules*, pada halaman ini juga terdapat *tab list* untuk melihat daftar *rule* pada setiap *file*.



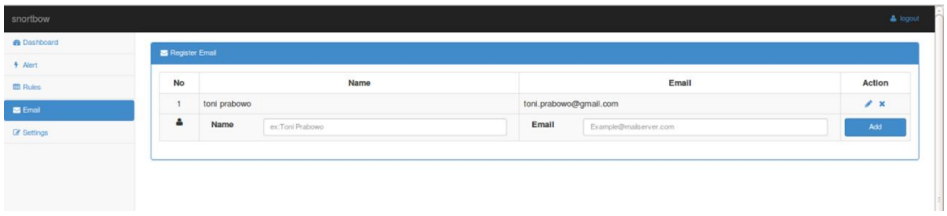
Gambar 4.14 Antarmuka Halaman Tambah dan Ubah Rule



Gambar 4.15 Antarmuka Halaman Daftar Rule

5. Halaman Email

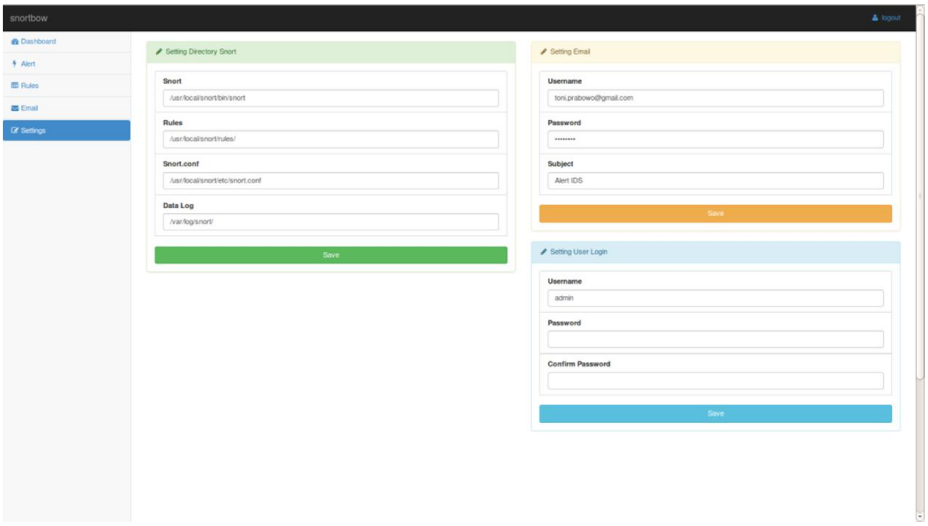
   Halaman ini digunakan untuk melihat, menghapus, mengubah dan menambah daftar *email* yang nantinya alamat *email* tersebut digunakan sebagai penerima *alert* yang dikirim oleh *server* IDS.



Gambar 4.16 Antarmuka Halaman Tambah, Ubah dan Hapus Email

6. Halaman Settings

   Halaman ini digunakan untuk melakukan konfigurasi terhadap aplikasi snortbow. Konfigurasi yang dapat dilakukan yaitu mengubah direktori dari *rules*, *binary* snort, data *log* snort dan *file* snort.conf. Selain mengubah direktori halaman ini juga digunakan untuk mengubah *user login* dan *user email*.



Gambar 4.17 Antarmuka Halaman Settings

4.2 Pengujian

Pengujian merupakan proses pengeksekusian dari implementasi yang telah dilakukan untuk menilai dan menemukan kesalahan-kesalahan yang terdapat pada sistem. Dengan adanya pengujian diharapkan sistem memilki kualitas yang baik dan dapat dijadikan bahan referensi dalam proses pengembangan sistem lebih lanjut.

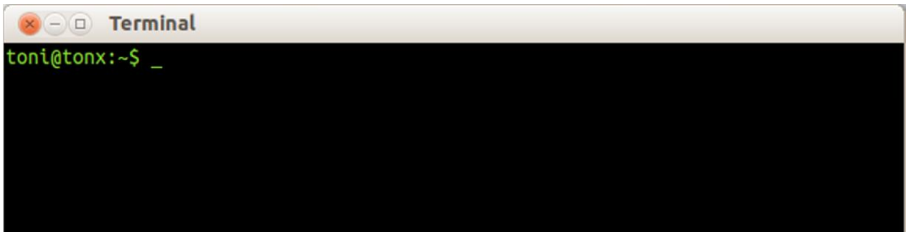
4.2.1 Pengujian Arsitektur Jaringan

Pengujian ini merupakan proses pengecekan terhadap keberhasilan dari instalasi arsitektur jaringan. Pengujian yang dilakukan yaitu dengan melakukan pengecekan terhadap status keterhubungan dari satu *node* ke *node* lain dan pengaksesan konten-konten yang ada di *server* oleh *client*.

4.2.1.1 Pengujian Keterhubungan Antar Perangkat

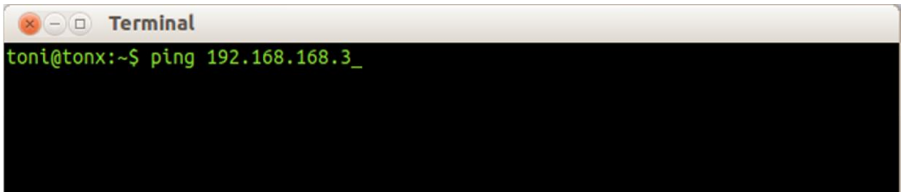
Pengujian ini merupakan pengujian yang dilakukan untuk mengecek setiap *node* atau perangkat dalam arsitektur jaringan saling terhubung. Pengujian dilakukan dengan menggunakan aplikasi ping. Berikut ini cara pengujian keterhubungan yang dilakukan dan hasil dari pengujian tersebut dijelaskan pada Gambar 4.18, Gambar 4.19 dan Gambar 4.20.

1. Membuka aplikasi *command prompt* pada windows atau *terminal* pada linux.



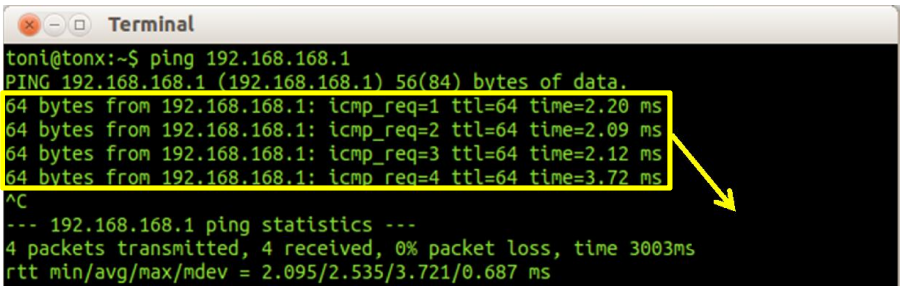
Gambar 4.18 Terminal Linux

2. Mengetikkan perintah ping <*ip address*>.



Gambar 4.19 Ping *IP Address*

3. Mengecek respon yang diberikan oleh *ip address* yang dituju.



Gambar 4.20 Status Respons Ping

Dari hasil pengujian yang dilakukan maka di dapatkan hasilnya yang dijelaskan pada Tabel 4.9.

Tabel 4.9 Hasil Pengujian Keterhubungan Perangkat

		Client					Server		IDS Server
		1	2	3	4	5	1	2	
Client	1	-	ok	ok	ok	ok	ok	ok	ok
	2	ok	-	ok	ok	ok	ok	ok	ok
	3	ok	ok	-	ok	ok	ok	ok	ok
	4	ok	ok	ok	-	ok	ok	ok	ok
	5	ok	ok	ok	ok	-	ok	ok	ok
Server	1	ok	ok	ok	ok	ok	-	ok	ok
	2	ok	ok	ok	ok	ok	ok	-	ok
IDS Server		ok	ok	ok	ok	ok	ok	ok	-

Pada Tabel 4.9 dapat disimpulkan bahwa setiap koneksi antar perangkat keras sudah saling terhubung.

4.2.2 Pengujian dan Penilaian IDS Snort

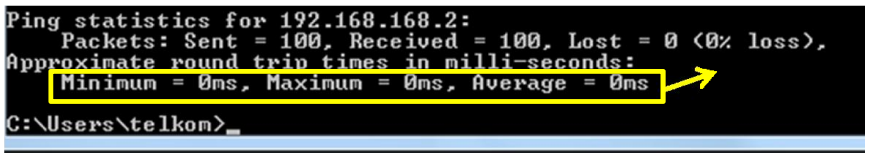
Pengujian ini merupakan proses untuk menguji aplikasi IDS snort yang terpasang pada sebuah jaringan komputer. Parameter yang diujikan yaitu kecepatan akses setelah IDS diterapkan pada jaringan, kecepatan itu antara komunikasi *client* ke *server* dan sebaliknya , kemudian parameter ketepatan dari IDS snort untuk mendeteksi jenis serangan. Berikut ini akan dijelaskn pengujian-pengujian tersebut.

4.2.2.1 Pengujian dan Penilaian Kecepatan Akses

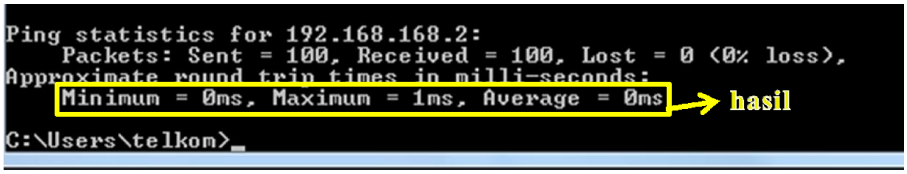
Pengujian ini merupakan proses untuk menguji dan menilai dari kecepatan akes komunikasi antara *client* dan *server* setelah IDS diterapkan, apakah setelah IDS terpasang mempengaruhi terhadap kecepatan akses. Untuk menguji parameter kecepatan ini dilakukan dengan mengisi *form* pada Tabel 4.10.

1. Pengujian Kecepatan Akses ke-1

   Pengujian ini dilakukan oleh *client* 1 (192.168.168.4) yang melakukan ping ke *web server* 1 (192.168.168.2) dengan menggunakan perintah pada *command prompt* yaitu `ping 192.168.168.2 -n 100`.



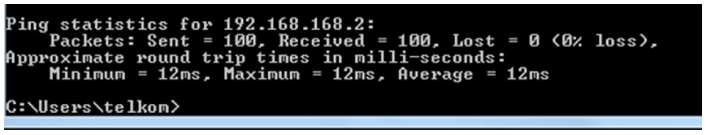
Gambar 4.21 Hasil Pengujian Ke-1 Sebelum Terpasang IDS



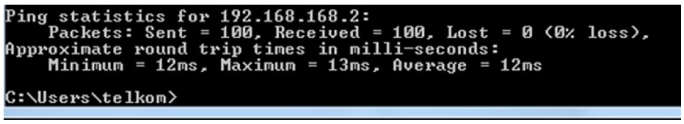
Gambar 4.22 Hasil Pengujian Ke-1 Setelah Terpasang IDS

2. Pengujian Kecepatan Akes Ke-2

   Pengujian ini dilakukan oleh *client* 1 (192.168.168.4) yang melakukan ping ke *web server* 1 (192.168.168.2) dengan menggunakan perintah pada *command prompt* yaitu `ping 192.168.168.2 -s 65000 -n 100`.



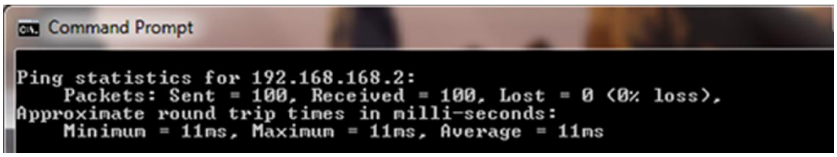
Gambar 4.23 Hasil Pengujian Ke-2 Sebelum Terpasang IDS



Gambar 4.24 Hasil Pengujian Ke-2 Setelah Terpasang IDS

3. Pengujian Kecepatan Akses Ke-3

   Pengujian ini dilakukan oleh *client* 4 dan *client* 5 yang melakukan ping ke *web server* 1 secara bersamaan dengan menggunakan perintah pada *command prompt* yaitu `ping 192.168.168.2 -l 65000 -n 100`.



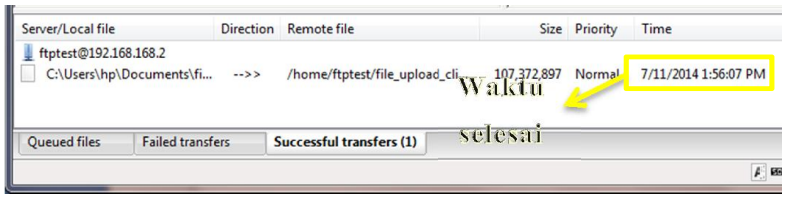
Gambar 4.25 Hasil Pengujian Ke-3 Sebelum Terpasang IDS



Gambar 4.26 Hasil Pengujian Ke-3 Setelah Terpasang IDS

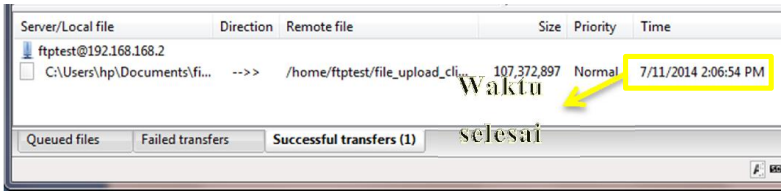
4. Pengujian Kecepatan Akses Ke-4

   Pengujian ini dilakukan oleh *client* 5 dengan menggunakan aplikasi Filezilla untuk meng-*upload file* dengan ukuran *file* 107 Mb ke *web server* 1. Waktu awal pada saat *upload file* yaitu 1:55:55.



Gambar 4.27 Hasil Pengujian Ke-4 Sebelum Terpasang IDS

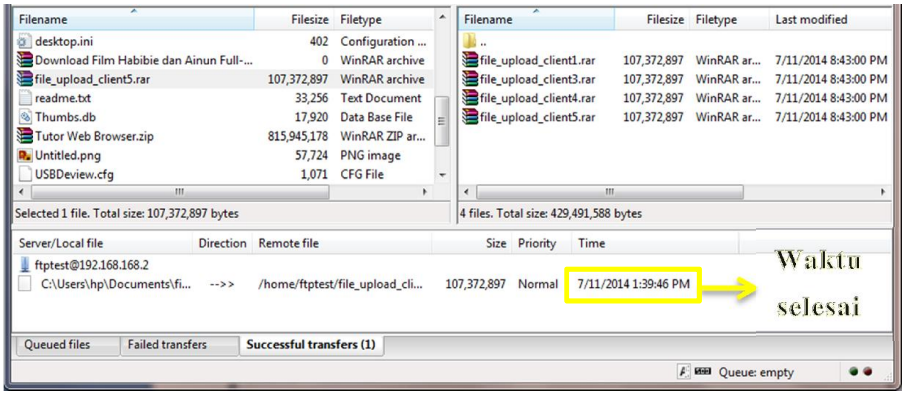
Pada pengujian ke-4 sebelum IDS terpasang proses *upload file* memakan waktu selama 12 detik. Waktu awal pada saat *upload file* setelah IDS terpasang yaitu 2:06:46 dan selesai proses *upload* terlihat pada Gambar 4.28, sehingga proses *upload file* setelah IDS terpasang memakan waktu selama 12 detik.



Gambar 4.28 Hasil Pengujian Ke-4 Setelah Terpasang IDS

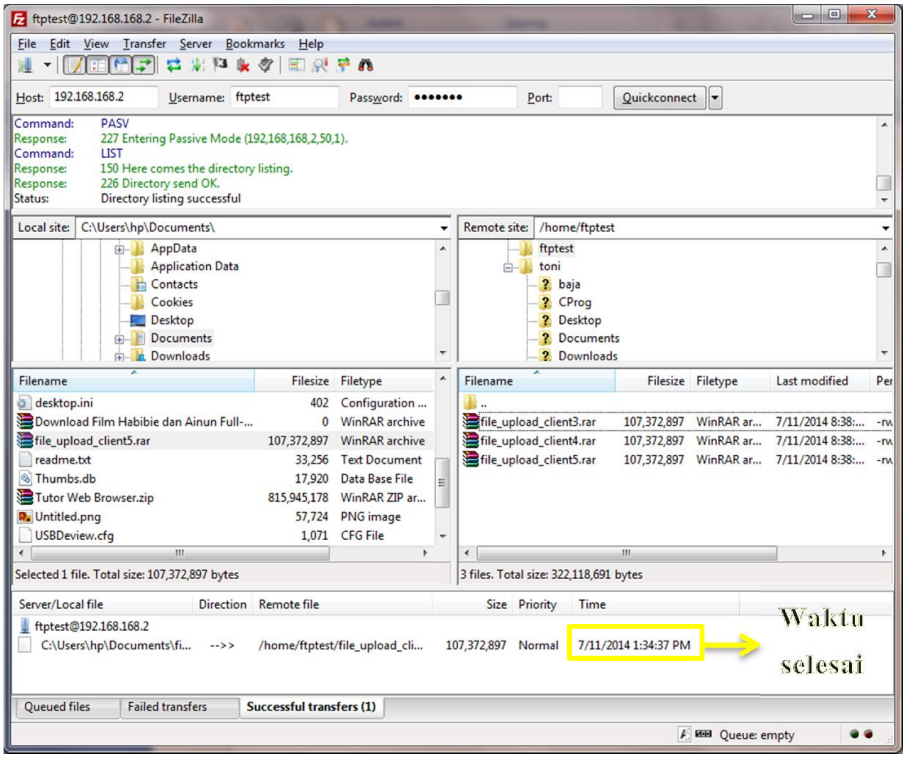
5. Pengujian Kecepatan Akses Ke-5

   Pengujian ini dilakukan oleh *client* 1,3,4,5 dengan menggunakan aplikasi Filezilla untuk meng-*upload file* dengan ukuran *file* 107 Mb ke *web server* 1 secara bersamaan. Waktu awal pada saat *upload file* yaitu 1:39:10.



Gambar 4.29 Hasil Pengujian Ke-5 Sebelum Terpasang IDS

Pada pengujian ke-4 sebelum IDS terpasang proses *upload file* memakan waktu selama 36 detik .Waktu awal pada saat *upload file* setelah IDS terpasang yaitu 1:34:01 dan selesai proses *upload* terlihat pada Gambar 4.30 , sehingga proses *upload file* setelah IDS terpasang memakan waktu selama 36 detik.



Gambar 4.30 Hasil Pengujian Ke-5 Setelah Terpasang IDS

6. Pengujian Kecepatan Akses Ke-6

   Pengujian ini dilakukan oleh *client* 1,3,4,5 dengan menggunakan aplikasi Filezilla untuk meng-*upload file* dengan ukuran *file* 107 Mb dan *client* 2 melakukan ping dengan paket 65000 bytes ke *web server* 1 secara bersamaan. Waktu awal eksekusi yaitu 1:51:48 dan selesai pada 1:52:35 sehingga waktu *upload file* pada pengujian ke-6 ini sebelum terpasang IDS yaitu 47 detik. Pengujian yang sama dilakukan pada saat IDS terpasang, waktu awal pengujian 1:59:32 dan selesai pada 2:00:19, sehingga waktu yang dibutuhkan untuk *upload file* pada pengujian ke-6 ini adalah 47 detik.

Tabel 4.10 Hasil Penilaian dari Proses Pengujian

No	Cara	Waktu (milidetik)	
		Sebelum Terpasang IDS	Setelah Terpasang IDS
1	1 Client Ping paket normal	0	0
2	1 Client Ping paket 65000 *bytes*	12	12
3	2 Client Ping paket 65000 bytes secara bersamaan.	11	11
4	1 Client Upload File Melalui FTP	1200	1200
5	4 Client Upload File Melalui FTP secara bersamaan.	3600	3600
6	1 *Client* Melakukan Ping paket 65000 *bytes*, 4 *Client* Melakukan *Upload file*. Secara bersamaan.	4700	4700

Dari hasil perhitungan kecepatan pada Tabel 4.10 dapat dihitung selisih dari waktu akses setelah dan sebelum terpasang IDS. Perhitungan selisih waktu dituliskan pada Tabel 4.11.

Tabel 4.11 Hasil Perhitungan Selisih Waktu Akses

No	Waktu sebelum terpasang IDS (mili detik)	Waktu setelah terpasang IDS (mili detik)	Selisih waktu (mili detik)
1	0	0	0
2	12	12	0
3	11	11	0
4	1200	1200	0
5	3600	3600	0
6	4700	4700	0

Dari hasil perhitungan pada Tabel 4.11 maka dapat terlihat selisih waktu akses antara setelah dan sebelum terapasang IDS yaitu 0 detik dengan kata lain penerapan IDS pada sebuah infrastruktur jaringan tidak berpengaruh terhadap kecepatan komunikasinya.

4.2.2.2 Pengujian dan Penilaian Ketepatan Deteksi Serangan

Pengujian ini merupakan proses untuk menguji ketepatan deteksi serangan yang dilakukan oleh aplikasi IDS. Pengujian dilakukan dengan jenis serangan SQL *injection*, XSS, *server scanning* dan DDOS yang dilakukan oleh *client* 1,2,3,4,dan 5 dalam waktu bersamaan. Proses pengujian dilakukan dengan mengisi form pada Tabel 4.12.

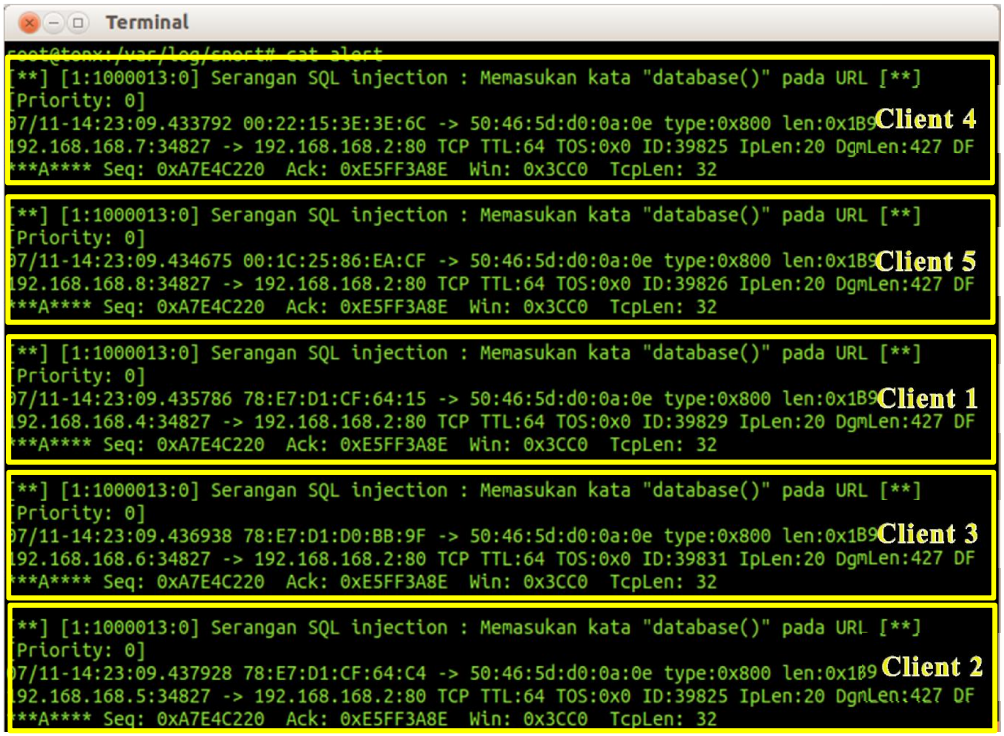
1. Pengujian Ketepatan Deteksi Ke-1

   Pengujian ini dilakukan menggunakan aplikasi firefox oleh setiap *client*. *Client-client* tersebut akan mengakses ke *web server* 1 dan melakukan serangan berupa SQL *injection*. Serangan yang dilakukan adalah dengan memasukkan url http://192.168.168.2/index.php?ask=kat&idkategori=-3+uni on+select+1,database(),3,4,version(),6,7,8,9,10,11--. Maka *web server* 1 akan menampilkan nama dari *database*-nya seperti pada Gambar 4.31.



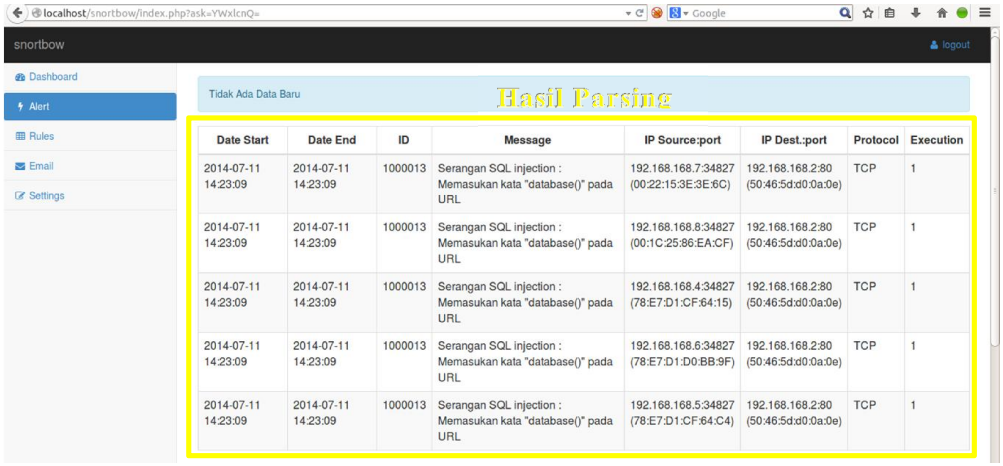
Gambar 4.31 Hasil Serangan Ke-1 SQL *Injection Client* 5

Dari pengujian dengan cara menyerang dengan teknik SQL *injection* oleh para *client*, didapatkan hasil deteksi ke-1 yang dijelaskan pada Gambar 4.32.



Gambar 4.32 Hasil Deteksi Ke-1 Pada *Terminal Server* IDS

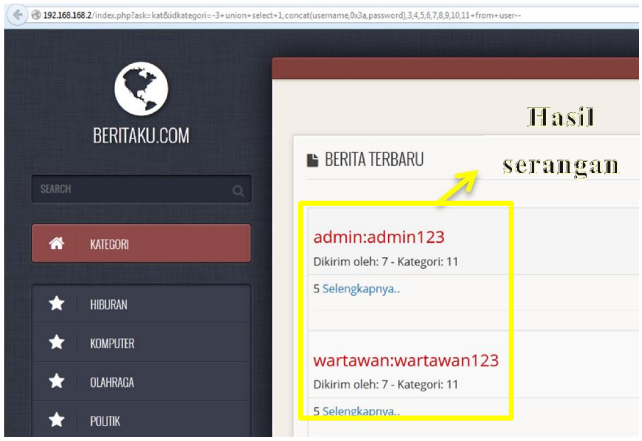
Pada Gambar 4.33 adalah hasil parsing *file* alert dari pengujian ke-1.



Gambar 4.33 Hasil Deteksi Ke-1 Pada Aplikasi Snortbow

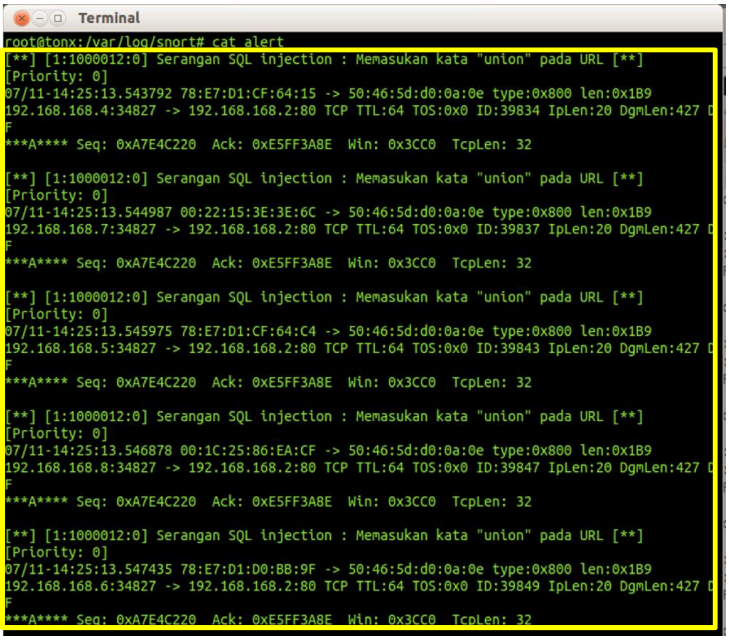
2. Pengujian Ketepatan Deteksi Ke-2

   Pengujian ini sama dengan pengujian ketepatan ke-1 hanya saja url yang dimasukkan yaitu http://192.168.168.2/index.php?ask=kat&idkategori=-3+union+select+1,concat(username,0x3a,password),3,4,5,6,7,8,9,10,11+from+user--.



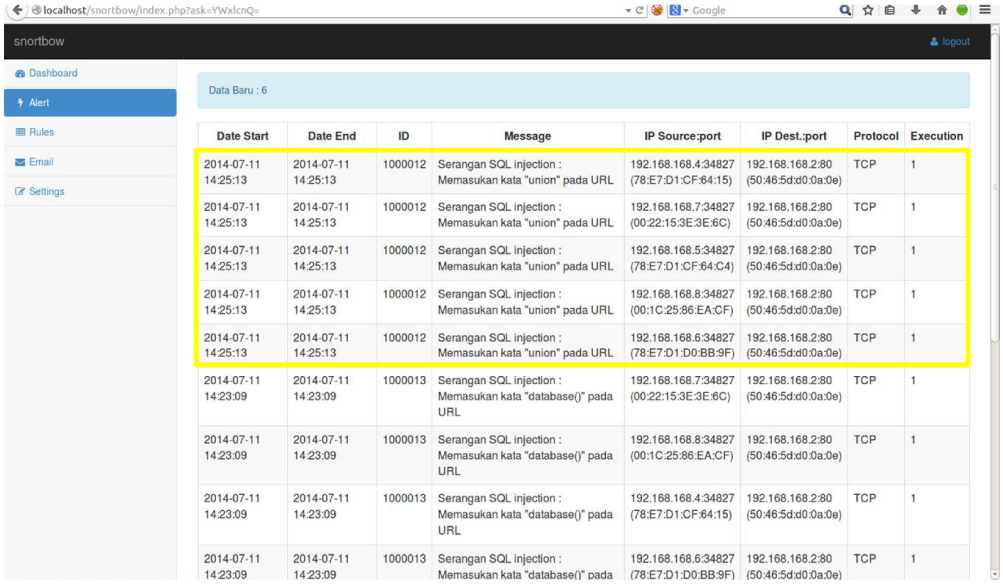
Gambar 4.34 Hasil Serangan Ke-2 SQL *Injection Client* 5

Dari pengujian dengan cara menyerang dengan teknik SQL *injection* oleh para *client*, didapatkan hasil deteksi ke-2 yang dijelaskan pada Gambar 4.35.



Gambar 4.35 Hasil Deteksi Ke-2 Pada *Terminal Server* IDS

Pada Gambar 4.36 adalah hasil parsing *file* alert dari pengujian ke-2.



Gambar 4.36 Hasil Deteksi Ke-2 Pada Aplikasi Snortbow

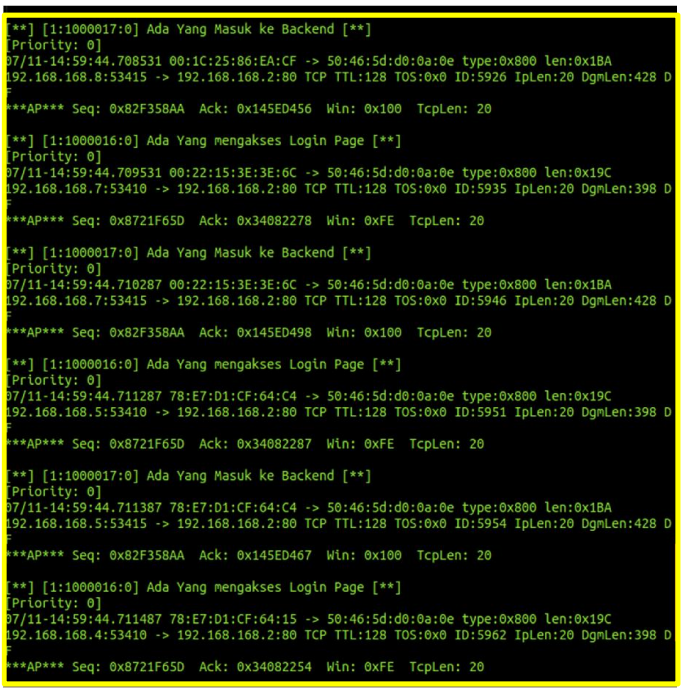
3. Pengujian Ketepatan Deteksi Ke-3

   Pengujian ini dilakukan dengan mengakses halaman login pada *web server*.



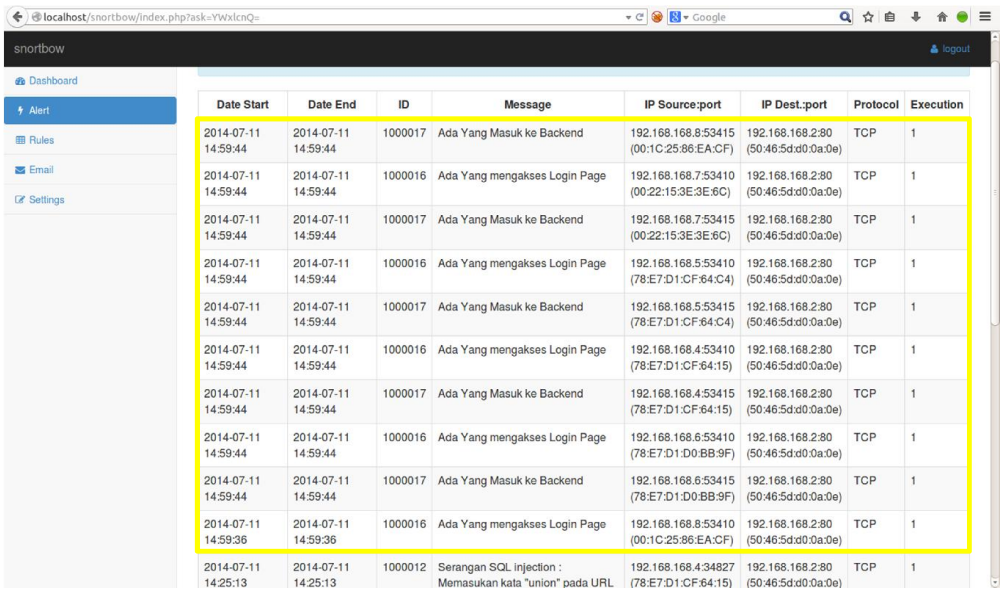
Gambar 4.37 Hasil Serangan Ke-3 SQL *Injection Client* 5

Dari pengujian dengan cara menyerang dengan teknik SQL *injection* oleh para *client*, didapatkan hasil deteksi ke-3 yang dijelaskan pada Gambar 4.38.



Gambar 4.38 Hasil Deteksi Ke-3 Pada *Terminal Server* IDS

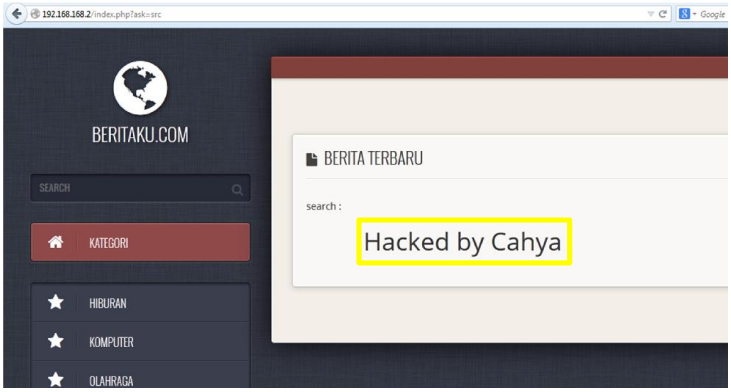
Pada Gambar 4.39 adalah hasil parsing file alert dari pengujian ke-3.



Gambar 4.39 Hasil Deteksi Ke-3 Pada Aplikasi Snortbow

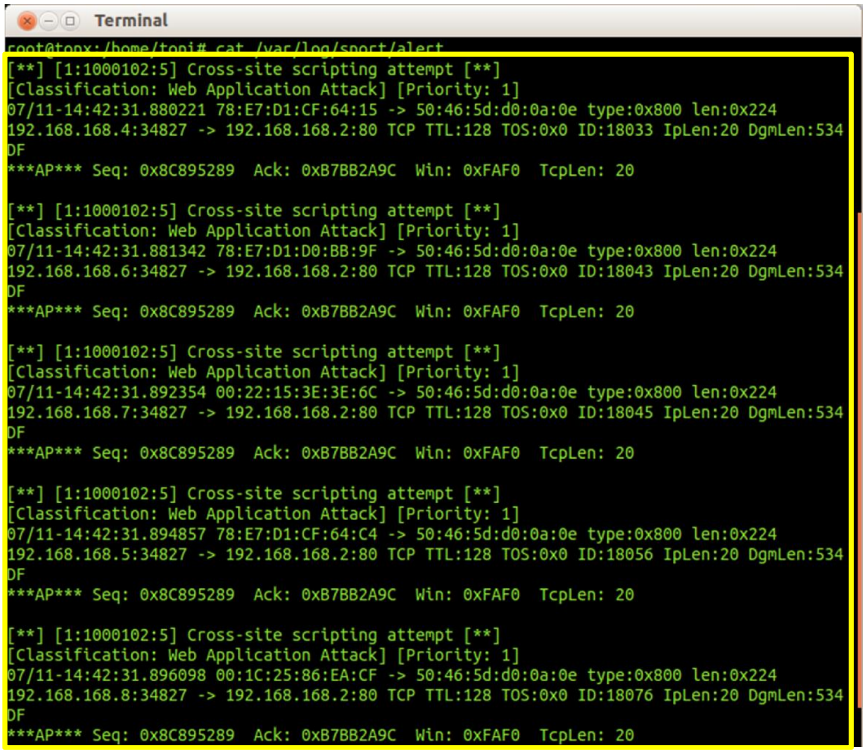
4. Pengujian Ketepatan Deteksi Ke-4

   Pengujian ini dilakukan *client* ke *web server* dengan memasukkan *tag* html pada kolom isian search. Hasil pengujian dapat dilihat pada Gambar 4.40.



Gambar 4.40 Hasil Serangan Ke-4 XSS *Client* 5

Dari pengujian dengan cara menyerang dengan teknik *cross site scripting* oleh para *client*, didapatkan hasil deteksi ke-4 yang dijelaskan pada Gambar 4.41.



Gambar 4.41 Hasil Deteksi Ke-4 Pada *Terminal Server* IDS

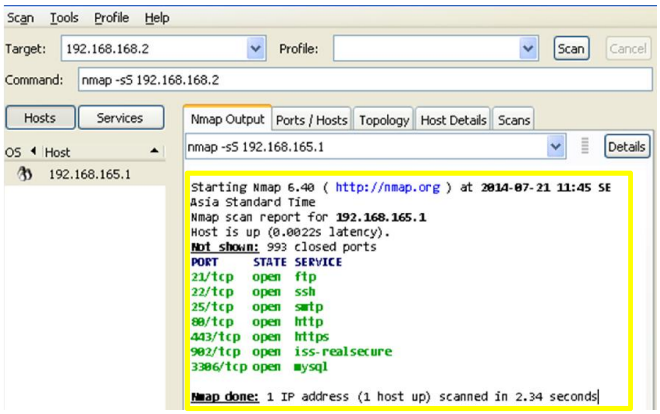
Pada Gambar 4.42 adalah hasil parsing *file* alert dari pengujian ke-4.



Gambar 4.42 Hasil Deteksi Ke-4 Pada Aplikasi Snortbow

5. Pengujian Ketepatan Deteksi Ke-5

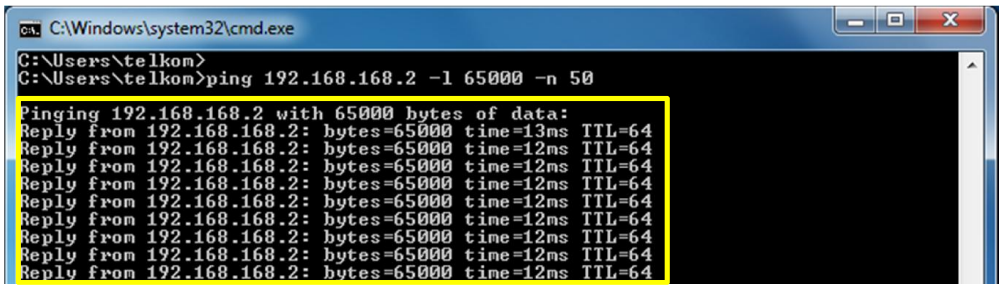
Pengujian ini dilakukan dengan menggunakan aplikasi zenmap, proses ini dilakukan oleh client 1,3,4 dan 5 terhapat *web server* 1. Proses ini adalah jenis metode serangan untuk mendapatkan informasi dari *server* yang akan diserang, informasi-informasi tersebut antara lain port, sistem operasi dan lain-lain. Berikut ini hasil serangan yang dilakukan yang digambarkan pada Gambar 4.43.



Gambar 4.43 Hasil Pengujian Ke-5 *Client* 5

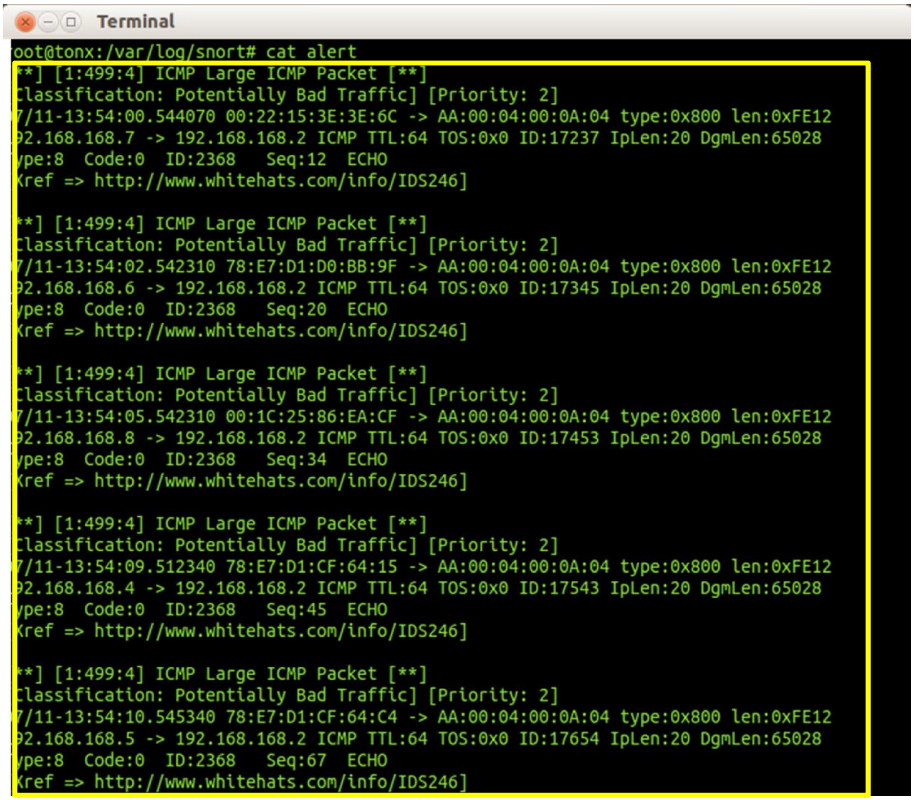
6. Pengujian Ketepatan Deteksi Ke-6

   Pengujian ini dilakukan oleh setiap *client* terhadap *web server* 1 dengan menggunakan aplikasi ping dengan paket data 65000 bytes secara bersamaan. Berikut ini digambarkan pada Gambar 4.44 adalah perintah command prompt yang dilakukan oleh *client* 5.



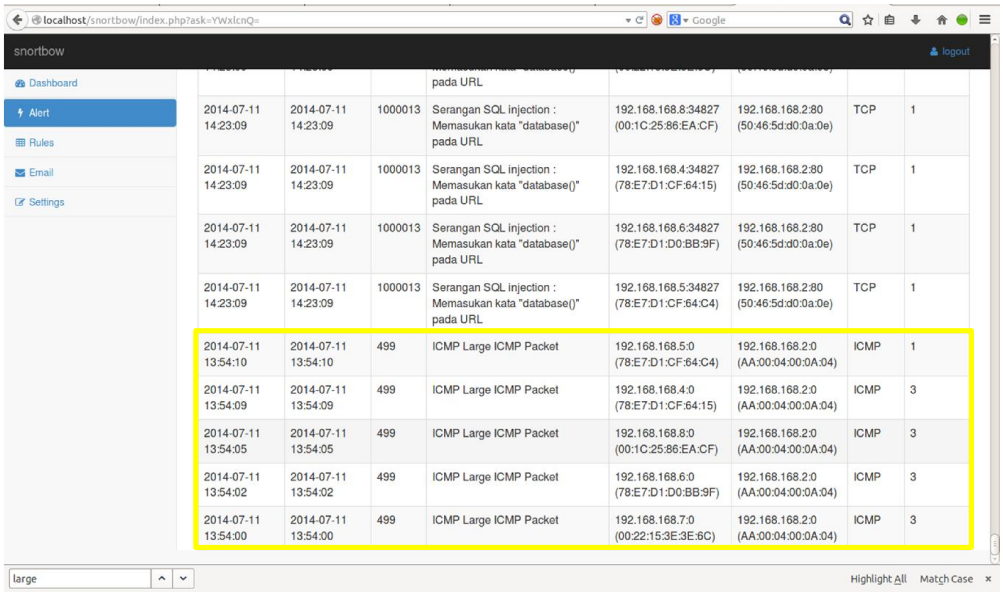
Gambar 4.44 Hasil Pengujian Ke-6 *Client* 5

Dari pengujian dengan cara menyerang dengan teknik *DoS* oleh para *client*, didapatkan hasil deteksi ke-6 yang dijelaskan pada Gambar 4.45.



Gambar 4.45 Hasil Deteksi Ke-6 Pada *Terminal Server* IDS

Pada Gambar 4.46 adalah hasil parsing *file* alert dari pengujian ke-6.



Gambar 4.46 Hasil Deteksi Ke-6 Pada Aplikasi Snortbow

Dari setiap serangan yang dilakukan oleh *client* terhadap *web server* akan dideteksi oleh *server* IDS. Berikut ini adalah hasil deteksi yang di-*log* oleh *server* IDS, hasil *log* tersebut yang digunakan untuk penilaian terhadap ketepatan deteksi dari IDS, dengan melihat apakah setiap *client* yang melakukan serangan terhadap *web server* terdapat *log* nya di *server* IDS.

Tabel 4.12 Pengujian Deteksi Serangan

No	Jenis Serangan	Status Deteksi Serangan Client				
		1	2	3	4	5
1	*SQL injection* : Mendapatkan nama database dari *web server*	T	T	T	T	T
2	*SQL injection* : Mendapatkan record-record dari *database*	T	T	T	T	T
3	*SQL injection* : Login ke webserver	T	T	T	T	T
4	*XSS :* Memasukkan tag html pada *web server*	T	T	T	T	T
5	*Server Scanning* : Menggunakan aplikasi nmap	T	T	T	T	T
6	*DDOS (Ping Large Data)*	T	T	T	T	T
Keterangan : T : Terdeteksi TD : Tidak Terdeteksi						

Dari Tabel 4.12 dapat disimpulkan bahwa untuk 5 *client*, server IDS masih mampu untuk mendeteksi jenis serangan SQL *injection*, XSS, *Port Scanning* dan DDOS (*Ping Large Data*).

4.3 Rekomendasi

Dari hasil analisa jenis serangan-serangan yang dimungkinkan untuk menyerang web server dan hasil analisa serta pengujian dari intrusion detection system maka terdapat beberapa rekomendasi untuk mencegah penyerangan

tersebut dan memaksimalkan kinerja IDS. Berikut ini merupakan rekomendasi yang diusulkan yaitu :

1. Dalam pembangunan aplikasi berbasis web maka harus melakukan validasi dari setiap input yang akan diproses.
2. Menutup port-port yang tidak diperlukan pada web server.
3. Membatasi besar paket yang lewat pada jaringan.
4. Mengelompokkan setiap rule pada sebuah file rules.