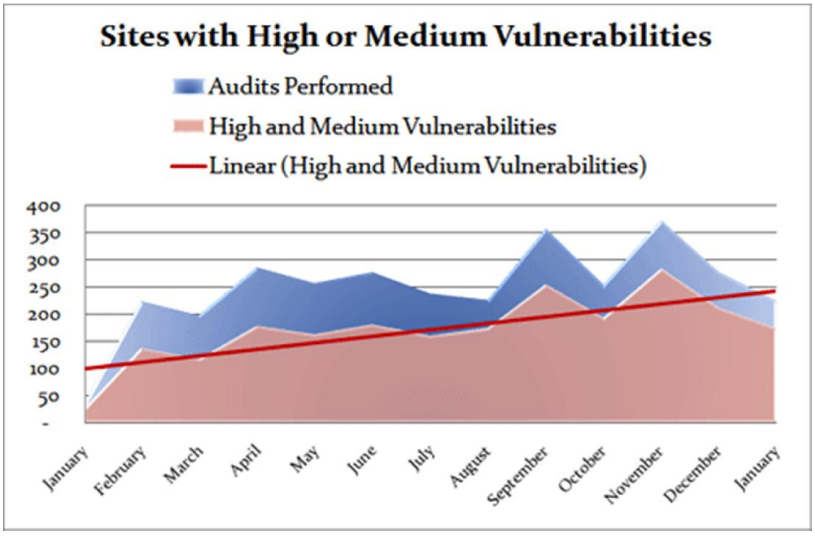
BAB 3
ANALISIS DAN PERANCANGAN

3.1 Analisis

Pada bagian ini akan dijelaskan mengenai penguraian dari masalah yang utuh ke dalam bagian-bagian komponen dengan maksud mengidentifikasi dan mengevaluasi permasalahan-permasalahan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan, sehingga dapat diusulkan perbaikan-perbaikannya.

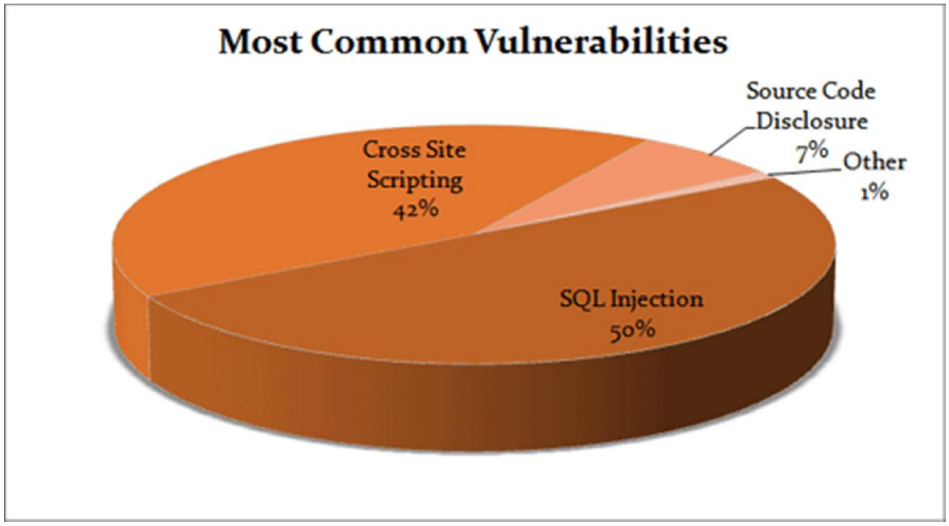
3.1.1 Analisis Masalah

Pada tahun 2006 sampai dengan 2007 acunetix melakukan *scanning* terhadap 10000 lebih *website*, dari *website-website* yang di *scan* hasilnya 70% dari *website* tersebut ditemukan kelemahan dengan tingkatan menengah dan tinggi. Kelemahan-kelemahan tersebut dapat ditemukan dan dimanipulasi oleh para *hacker* untuk mencuri data-data yang sensitif. Hasil *scanning* dapat dilihat pada Gambar 3.1, dari grafik tersebut hasilnya cenderung meningkat setiap bulannya.



Gambar 3.1 Grafik Hasil *Scan Website* Oleh Acunetix

Diantara *website-website* tersebut kelemahan yang paling banyak adalah *website* yang dapat diserang dengan menggunakan jenis serangan *SQL injection* dan *XSS* (*Cross Site Scripting*).



Gambar 3.2 Persentase Vulnerabilities

SQL injection dan *XSS* adalah metode serangan yang dilakukan untuk mendapatkan suatu informasi dari suatu *website*. Penyebab suatu *website* memiliki celah untuk diserang *SQL injection* dan *XSS* adalah kurangnya metode *input encoding* dalam proses penyaringan karakter sehingga mengizinkan karakter tertentu untuk dieksekusi. Kedua ancaman ini terjadi pada layer aplikasi karena celah ini muncul dari aplikasi yang disimpan pada *web server*.

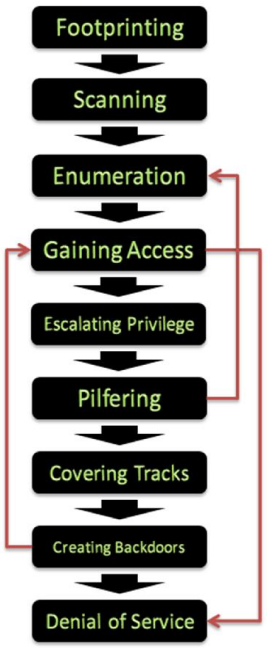
Selain *SQL injection* dan *XSS* suatu *server web* dapat dimungkinkan diserang dengan menggunakan beberapa metode serangan, pada saat ini metode serangan yang paling banyak dilakukan adalah jenis serangan *DDOS* (*Distributed Denial of Sevice*). Serangan jenis ini sulit dihalau karena pada umumnya melakukan blokir pada suatu alamat IP tidak akan menghentikan serangan. *Traffic* berjalan mulai dari ratusan atau sampai ribuan jumlahnya. Serangan ini meliputi :

a. *FTP Bounce Attacks*
b. *Port Scanning Attack*
c. *Ping Flooding Attack*

d. *Smurf Attack*
e. *SYN Flooding Attack*
f. *IP Fragmentation/Overlapping Fragment Attack*
g. *IP Sequence Prediction Attack*
h. *DNS Cache Poisoning*
i. *SNMP Attack*
j. *Send Mail Attack*

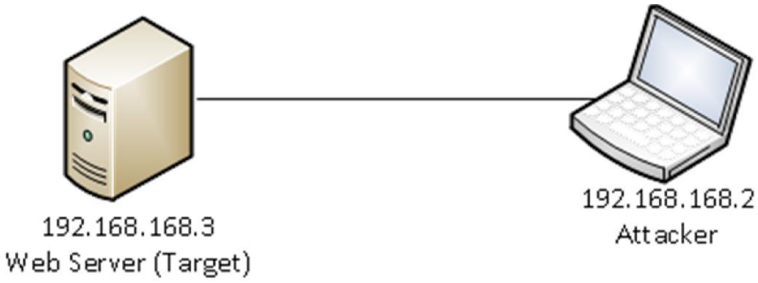
3.1.2 Analisis Metode Serangan

Dari permasalahan diatas, ada banyak metode-metode serangan terhadap *web server* yang mungkin dilakukan untuk membuat sebuah *server down* atau tidak dapat diakses. Serangan yang paling fatal adalah serangan yang dapat menghancurkan *hardware* dari *server* tersebut. Dalam melakukan penyerangan terhadap *server* maka penyerang (*attacker*) melakukan beberapa langkah, langkah-langkah tersebut merupakan anatomi dalam melakukan *hacking*. Masing-masing langkah memiliki tujuan tersendiri sehingga setelah langkah pertama tercapai tujuannya maka langkah selanjutnya dapat dilakukan.



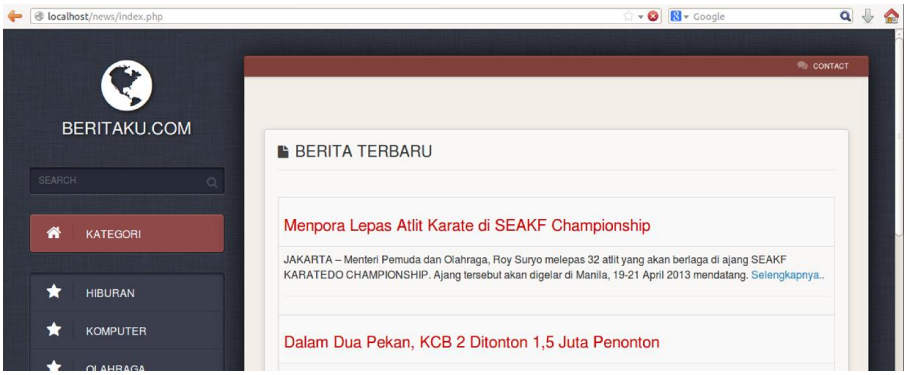
Gambar 3.3 *Hacking Anatomy*

Untuk setiap langkah-langkah serangan tersebut memiliki beberapa ciri-ciri yang dapat dideteksi, ciri-ciri tersebut dapat dilihat dari jenis paket yang dikirimkan maupun dari pola penyerangannya. Dalam penelitian ini dilakukan beberapa analisa mengenai metode penyerangan terhadap *web server*. Proses analisa penyerangan tersebut dilakukan pada skema jaringan seperti pada Gambar 3.4 berikut ini :



Gambar 3.4 Skema Jaringan Analisa Penyerangan

Pada *web server* terdapat aplikasi *web based* yang digunakan untuk memberikan informasi mengenai berita.



Gambar 3.5 Tampilan *Web* Pada *Web Server*

Dalam skema diatas dilakukan beberapa percobaan penyerangan. Berikut ini akan dianalisa bagaimana serangan tersebut dapat menggangu layanan dari suatu target. Pada *Web Server* paket data dari penyerang akan di analisa menggunakan aplikasi *wireshark* versi 1.6.7.

3.1.2.1 Analisis *Footprinting*

Proses dalam melakukan penyerangan atau *hacking* pada sebuah komputer atau *server* diawali dengan melakukan teknik *footprinting*. *Footprinting* dalam dunia *hacking* adalah segala kegiatan untuk mengumpulkan semua informasi dari target yang akan diserang, informasi – informasi tersebut diantaranya sistem operasi, *ip address* , aplikasi, dan informasi-informasi penting sebelum melakukan penyerangan yang dimiliki target. Proses *footprinting* bukan merupakan tindakan ilegal, proses ini di ibaratkan orang yang sedang mencari biodata seseorang yang sudah dipublikasikan. Dari informasi-informasi tersebut *attacker* dapat mengetahui kelemahan dari komputer atau *server* tersebut. Proses *footprinting* dapat dilakukan dengan dua cara yaitu :

1. *Inner Footprinting*
   Mencari informasi *server* dimana *attacker* sudah berada di dalam jaringan *server* target.
2. *Outer Footprinting*
   Mencari informasi *server* dimana *attacker* berada di luar jaringan *server* target.

Dalam penelitian ini cara yang digunakan *inner footprinting*, sesuai dengan skema pada Gambar 3.4 *attacker* berada di jaringan yang sama dengan *server*. Informasi yang didapatkan yaitu ip address 192.168.168.3, sistem operasi linux ubuntu, aplikasi web server apache.

3.1.2.2 Analisis Scanning

Pada dasarnya proses *scanning* dapat juga dikategorikan sebagai proses *footprinting*. Perbedaan dalam proses *scanning* dan *footprinting* yaitu pada proses *scanning attacker* melakukan pencarian informasi lebih detail dengan memetakan kelemahan dan kekuatan target. Proses *scanning* ini sudah lebih intensif dari sekedar mencari informasi atau dengan kata lain sudah ada beberapa tindakan destruktif ringan yang terjadi pada target.

Proses *scanning* yang dilakukan *attacker* menggunakan aplikasi nmap. Pertama yang dilakukan adalah untuk mendapatkan port-port yang terbuka pada server dengan cara command *"nmap 192.168.168.3 -sS"*, *command* ini dilakukan untuk men-*scan port* pada IP 192.168.168.3 dengan teknik *SYN*.

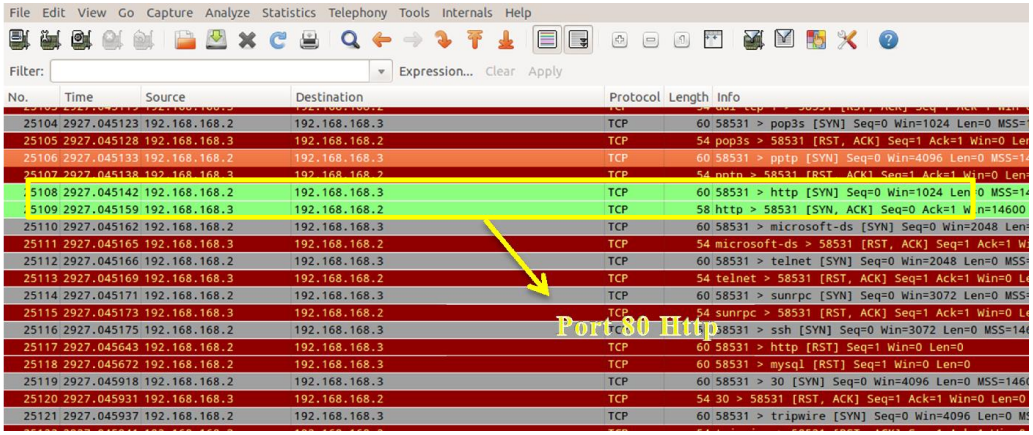
```
Terminal
root@tonx:/home/toni# nmap 192.168.168.3 -sS

Starting Nmap 5.21 ( http://nmap.org ) at 2014-04-19 00:27 WIB
Nmap scan report for 192.168.168.3
Host is up (0.0000050s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
902/tcp  open  iss-realsecure
3306/tcp open  mysql

Nmap done: 1 IP address (1 host up) scanned in 13.07 seconds
```

Gambar 3.6 Hasil Scanning Port

Dari hasil *scanning port* didapat empat *port* yang terbuka yaitu http(80), https(443), iss-realsecure(902), mysql(3306). Di bawah ini adalah gambar paket data yang dideteksi oleh web server melalui aplikasi *wireshark.*



Gambar 3.7 Hasil Monitor Paket Data *Scanning Port*

Terlihat pada Gambar 3.7 ada proses *SYN* dan *ACK*, ini adalah cara *scanning port* dengan menggunakan teknik *SYN* yaitu *attacker* mengirimkan paket *SYN* ke *port* yang akan di cek, jika ada *reply SYN ACK* maka *port* tersebut terbuka. Paket data pada saat proses ini dijelaskan pada Tabel 3.1.

Tabel 3.1 Pengiriman Paket Data Scanning Port

Pengirim	Paket Data
192.168.168.2 (*attacker*)	aa 00 04 00 0a 04 00 26 2d 83 61 26 08 00 45 00 00 2c 32 5e 00 00 2c 06 8b 17 c0 a8 a8 02 c0 a8 a8 03 e4 a3 00 50 4b 4d 87 77 00 00 00 00 60 02 04 00 0b 17 00 00 02 04 05 b4 00 00
192.168.168.3 (*web server*)	00 26 2d 83 61 26 aa 00 04 00 0a 04 08 00 45 00 00 2c 00 00 40 00 40 06 69 75 c0 a8 a8 03 c0 a8 a8 02 00 50 e4 a3 62 e7 c2 c5 4b 4d 87 78 60 12 39 08 d1 75 00 00 02 04 05 b4

Penjelasan untuk paket-paket data diatas dapat dilihat pada Tabel 3.2 dan

Tabel 3.3.

Tabel 3.2 Penjelasan paket data dari *Attacker*

Data	Penjelasan
aa 00 04 00 0a 04	Mac address destination (*WebServer*) aa:00:04:00:0a:04
00 26 2d 83 61 26	Mac address Source (*Attacker*) 00:26:2d:83:61:26
08 00	Header IP
45	Header Length 20 bytes
00 2c	Total Length 44 bytes
32 5e	Identification 0x325e (12895)
00 00	Fragment Offset : 0
2c	Time to live : 44
06	Protocol TCP

Data	Penjelasan
8b 17	Checksum
c0 a8 a8 02	IP address source (Attacker) 192.168.168.2
c0 a8 a8 03	IP address destination (WebServer) 192.168.168.3
e4 a3 00 50 4b 4d 87 77 00 00 00 00 60 02 04 00 0b 17 00 00 02 04 05 b4	Port source (Attacker) e4 a3 : 58531 Port destination (WebServer) 00 50 : 80 Seq : 0, Len:0 Flag : 0x002 (SYN)

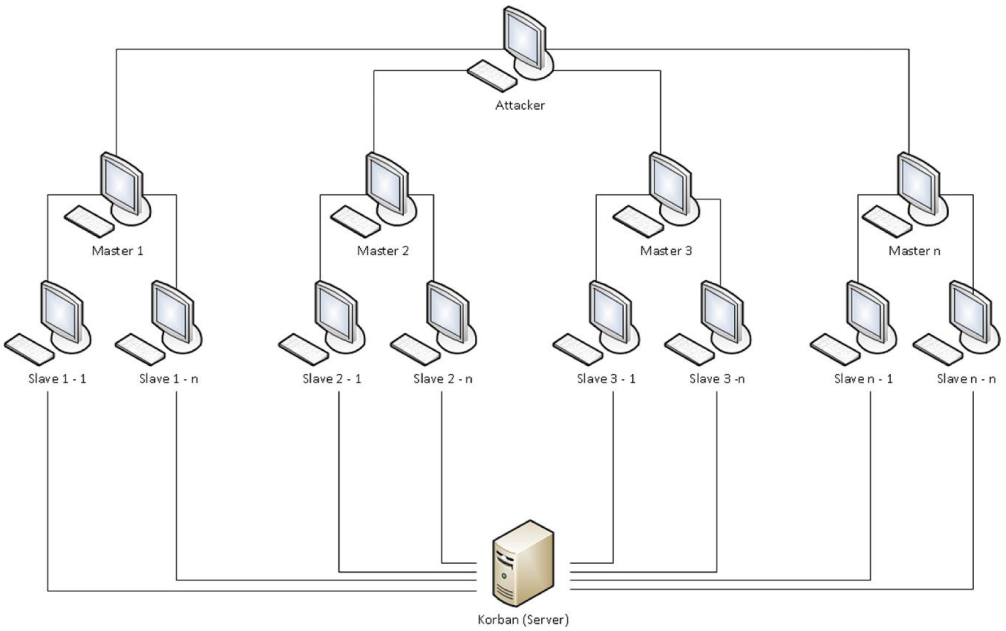
Tabel 3.3 Penjelasan paket data dari *Web Server*

Data	Penjelasan
00 26 2d 83 61 26	Mac address destination (*Attacker*) 00:26:2d:83:61:26
aa 00 04 00 0a 04	Mac address source (*WebServer*) aa:00:04:00:0a:04
08 00	Header IP
45	Header Length 20 bytes
00 2c	Total Length 44 bytes
00 00	Identification 0x0000 (0)
40 00	Fragment Offset : 0
40	Time to live : 64
06	Protocol TCP
69 75	Checksum
c0 a8 a8 03	IP address source (WebServer) 192.168.168.3
c0 a8 a8 02	IP address destination (Attacker) 192.168.168.2
00 50 e4 a3 62 e7 c2 c5 4b 4d 87 78 60 12 39 08 d1 75 00 00 02 04 05 b4	Port source (WebServer) 00 50 : 80 Port destination (Attacker) e4 a3 : 58531 Seq : 0, Len:0,Ack: 1, Flag : 0x012 (SYN, ACK)

3.1.2.3 Analisis serangan *Distributed Denial Of Service* (DDOS)

DOS (*Denial Of Service*) merupakan serangan yang dilancarkan melalui paket-paket jaringan tertentu, biasanya paket-paket sederhana dengan jumlah yang sangat besar dengan maksud mengacaukan keadaan jaringan target. Pada serangan *Distributed* DOS (DDOS), penyerang melakukan instalasi suatu *agent* atau *daemon* pada beberapa *host* yang telah berhasil dimasuki. Penyerang mengirimkan perintah pada bagian *master*, yang mengakibatkan terkirimnya perintah pada beberapa *host slave*. *Master* melakukan komunikasi kepada *agent* yang berada pada *server* lain untuk melakukan perintah serangan.

Secara sederhana serangan *DDOS* dapat dilakukan dengan menggunakan perintah *ping*. Proses *ping* ini ditujukan kepada *sever* yang akan menjadi korban. Jika perintah ini hanya dilakukan oleh sebuah komputer mungkin tidak menimbulkan efek berarti bagi *server*. Namun jika dilakukan oleh banyak komputer maka perintah ini dapat memperlambat kinerja dari *server* tersebut.



Gambar 3.8 Skema Penyerangan DDOS

Pada Gambar 3.8 komputer *attacker* melakukan perintah kepada komputer *master* untuk menyerang *server*, jika satu komputer mengirimkan data 32

bytes/detik ke *server* yang dituju. Misalkan ada 10000 komputer yang melakukan perintah tersebut secara bersamaan, maka kiriman data sebesar 312 Mb/detik. Selanjutnya *server* tersebut secara langsung akan merespons kiriman yang dikirim dari 10000 komputer secara bersamaan. Jika 312 Mb/detik data yang harus diproses *server* , maka dalam 1 menit *server* harus memproses kiriman data sebesar 312 Mb x 60 detik = 18720 Mb.

Serangan *DDOS* ini umumnya bertujuan menghabiskan *bandwidth* dan membuat *crash* sistem sehingga tidak dapat memberikan pelayanan. Contohnya sebuah *server* perusahaan yang inti bisnisnya berorientasi pada *internet* tidak bisa memberikan pelayanan kepada *customer*-nya akibat kehabisan *bandwidth* selama beberapa jam atau bahkan berhari-hari, bisa dipastikan perusahaan tersebut akan mengalami kerugian secara materi. Selain dengan menggunakan perintah *ping*, terdapat beberapa jenis serangan *DDOS* lain diantaranya :

a. *Syn Flooding Attack*

Serangan *syn flooding* dilakukan dengan cara memanfaatkan kelemahan protokol pada saat terjadinya proses *handshake*. Saat dua komputer memutuskan untuk memulai melakukan komunikasi, komputer pengirim (*attacker*) akan mengirimkan *syn*, penerima (target) akan menjawab dengan mengirimkan *syn ack* kepada komputer pengirim. Seharusnya setelah menerima balasan *syn ack* dari penerima, pengirim mengirimkan *ack* kepada penerima untuk melakukan proses *handshake*. Namun pada kenyataannya, pengirim justru mengirimkan banyak paket *syn* kepada penerima yang mengakibatkan penerima harus terus menjawab permintaan dari pengirim. *Syn flooding attack* dalam penelitian ini menggunakan skema pada Gambar 3.4 untuk menganalisa paket data dan sebab akibat yang ditimbulkan dari jenis serangan *syn flooding attack*.

Dalam penelitian ini dilakukan dua metode *syn attack*, *syn attack* yang digabung dengan *ip spoofing* dan tidak menggunakan *ip spoofing* (Pemalsuan IP *address*). *Syn flooding attack* dilakukan dengan menggunakan aplikasi *hping3* pada sistem operasi *ubuntu*. Berikut ini adalah hasil analisa *syn flooding attack* yang dilakukan.

```
root@tonx:~# hping3 -a 192.168.168.100 -S 192.168.168.3 --flood -s 80 --keep
HPING 192.168.168.3 (eth0 192.168.168.3): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

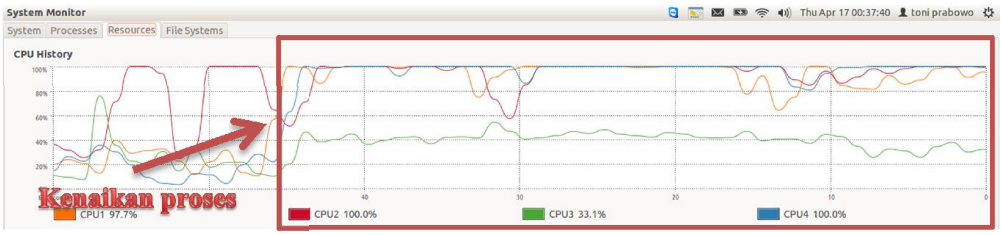
Gambar 3.9 *Command* SYN *Flooding Attack* Hping3

Komputer penyerang menggunakan aplikasi *hping3* dengan *command* seperti gambar Gambar 3.9, keterangan *command* tersebut yaitu :

Tabel 3.4 Penjelasan Parameter Hping3

Parameter	Fungsi
Hping3	Aplikasi hping3
-a	Parameter untuk membuat alamat palsu
192.168.168.100	Alamat palsu yang digunakan
-S	Parameter untuk mengirim Syn
192.168.168.3	Ip address target
--flood	Mengirimkan paket sebanyak dan secepat mungkin
-s	Parameter untuk menentukan baseport
80	Port yang akan menjadi jalur penyerangan
--keep	Tetap menyerang baseport

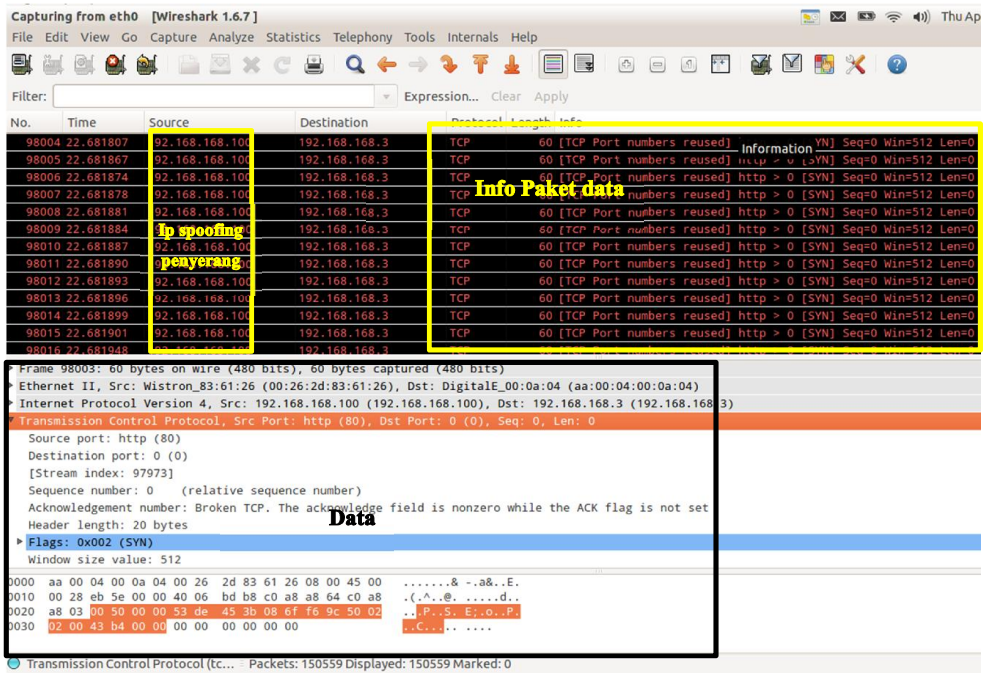
Setelah dilakukan penyerangan menggunakan *syn flooding*, terjadi kenaikan CPU proses pada komputer target. Terlihat pada Gambar 3.7 Proses CPU dari komputer target pada saat sebelum dan sesudah penyerangan menggunakan *sync flooding attack*.



Gambar 3.10 Kenaikan Proses CPU komputer Target

Kenaikan CPU proses ini dapat menyebabkan *processor* pada komputer menjadi lebih panas, jika proses ini berjalan dengan waktu yang lama dimungkinkan komputer target akan *crash* atau *hang* dan mengakibatkan

komputer *reboot*. Terlihat pada Gambar 3.11 merupakan hasil *monitoring* paket data ketika dilakukan *syn flooding attack* oleh komputer penyerang.



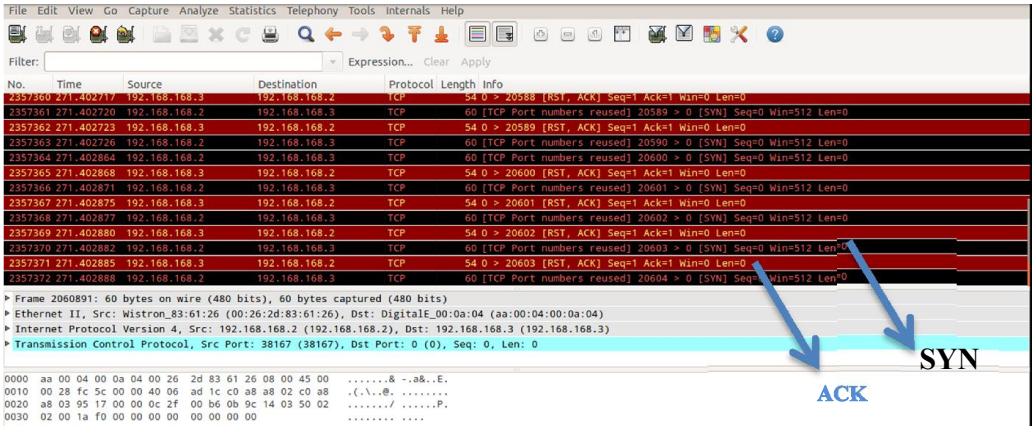
Gambar 3.11 Monitoring Network Pada Komputer Target

Pada Gambar 3.11 telihat pada kolom *source* tertulis 192.168.168.100, ini adalah *ip address* penyerang yang telah di palsukan, alamat penyerang seharusnya 192.168.168.2. Pada kolom info terdapat keterangan mengenai informasi paket data, terlihat komputer target tidak mengirimkan SYN ACK karena *ip address* penyerang telah dipalsukan dengan *ip address* 192.168.168.100 yang tidak ada dalam jaringan. Paket data yang dikirim oleh komputer penyerang yaitu aa 00 04 00 0a 04 00 26 2d 83 61 26 08 00 45 00 00 10 00 28 01 dc 00 00 40 06 a7 3b c0 a8 a8 64 c0 a8 00 20 a8 03 00 50 00 00 5d d9 f6 37 43 8a 76 ac 50 02 0030 02 00 cd 91 00 00 00 00 00 00 00 00. Data tersebut memiliki keterangan yang akan dijelaskan pada Tabel 3.5:

Tabel 3.5 Penjelasan Paket Data

Data	Keterangan
aa 00 04 00 0a 04	Mac Address komputer target (aa:00:04:00:0a:04)
00 26 2d 83 61 26	Mac Address komputer penyerang (00:26:2d:83:61:26)
08 00 45	Type IP Versi 4 Header length 20 byte
00	Differentiated Service Field
00 28	Total Length 40
01 dc	Identification 476
00 00	Fragment offset : 0
40	TTL : 64
06	Protocol : TCP
A7 3b	Checksum 0xa73b
c0 a8 a8 64	Ip address penyerang : 192.168.168.100
c0 a8 a8 03	Ip Address Target : 192.168.168.3
00 50	Source port 80 (http)
00 00	Destination port 0
5d d9 f6 37	Squence Number
43 8a 76 ac	Acknowledgement number : Broken TCP
50 02	Header Length : 20 bytes Flags 0x002 (SYN)
02 00	Window size value : 512
Cd 91	Checksum
00 00 00 00 00 00	Trailer

Berbeda jika *syn flooding attack* dilakukan tanpa menggunakan *ip spoofing*, maka hasilnya dapat dilihat pada Gambar 3.12.



Gambar 3.12 Paket Data SYN *Flooding Attack* Tanpa IP *Spoofing*

Pada Gambar 3.12 telihat ketika komputer penyerang melakukan *SYN* maka komputer target merespon dan mengirimkan *ACK* ke komputer penyerang. Perbedaan penggunaan *ip spoofing* dan yang tidak menggunakan *ip spoofing* akan berdampak pada penggunaan *bandwidth* pada komputer target. Ketika komputer penyerang tidak memalsukan *ip address*-nya maka komputer target akan mengirimkan *ACK* kepada komputer penyerang sejumlah paket *SYN* yang diterima. Pada Gambar 3.13 adalah sistem monitor komputer target dengan dua jenis penyerangan yaitu penyerangan *syn flooding attack* dengan *ip spoofing* dan *syn flooding attack* tanpa *ip spoofing*. Terlihat pada *network history process send* dan *receive*, pada awal penyerangan dilakukan syn *flooding attack* tanpa ip *spoofing* sehingga grafik *send* dan *receive* sejajar, setelah beberapa detik dilakukan *syn flooding menggunakan ip spoofing* sehingga grafik dari *receiving* lebih besar daripada grafik *send*.



Gambar 3.13 *System Monitor* Komputer Target

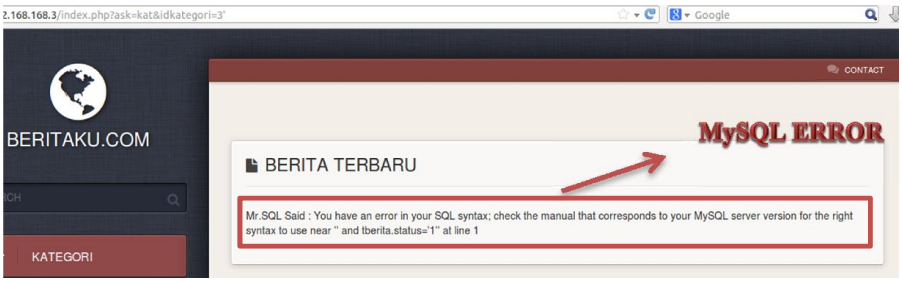
Hasil analisa dari penyerangan *syn flooding attack* yaitu serangan ini dapat berdampak sangat fatal kepada komputer target apabila komputer penyerang sangat banyak jumlahnya. Dengan penyerangan yang dilakukan oleh banyak penyerang proses CPU pada komputer target akan mengalami kenaikan yang signifikan yang dapat mengakibatkan komputer target tidak dapat melayani *request* dari komputer lain.

3.1.2.4 SQL *injection*

SQL *injection* merupakan salah satu metode serangan yang terjadi pada *web server*, metode ini dapat mengizinkan pengguna mengeksekusi *database* melalui kelemahan aplikasi *web* untuk memperoleh informasi penting. Mekanisme penyerangannya yaitu dilakukan dengan memanfaatkan kesalahan pada kode program yang tidak di filter. Pada penelitian metode ini akan dianalisa menggunakan skema pada Gambar 3.4, pada *web server* telah ada aplikasi web untuk memberikan informasi mengenai berita. Langkah – langkah dalam memulai SQL *injection* yaitu :

a. Mencari Celah Keamanan

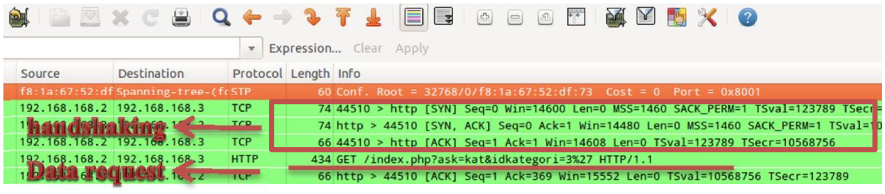
Mekanisme ini dilakukan pada URL yang biasanya terdapat parameter yang dikirim melalui metode GET. Pada kasus ini *attacker* mendapatkan url http://192.168.168.3/index.php?ask=kat&idkategori=3. Untuk mencari celah keamanan adalah dengan menambahkan tanda petik ‘ di belakang nilai dari parameter, menjadi http://192.168.168.3/index.php?ask=kat&idkategori=3’. Maka *website* akan merespon kesalahan yaitu *“You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' and tberita.status='1'' at line 1”.*



Gambar 3.14 Pesan *Error* MySQL

Dari celah ini dapat diketahui bahwa *web server* ini menggunakan MySQL *server* sebagai *database server*-nya. Penambahan tanda petik itu juga untuk mengetahui bahwa parameter pada URL berpengaruh pada *query sql*. Maka dapat diprediksi SQL *query* yang digunakan yaitu *“SELECT * from tberita where id=3”,* ketika menambahkan tanda petik maka akan terjadi *query* error. Dapat disimpulkan bahwa variable id tidak di filter.

Web server dapat mendeteksi proses ini, pada kasus ini pendeteksian paket data menggunakan *wireshark* untuk melihat paket.



Gambar 3.15 Paket Data Mencari Celah SQL *injection*

Pada Gambar 3.15 telihat proses *request* dan *reply* dari *attacker* dan *web server*. Diawali dengan *attacker* mengirimkan paket SYN dan di reply oleh *web server* dengan SYN,ACK lalu *attacker* mengirimkan ACK dan data yang di *request* ke *web server* melalui protokol HTTP (*port* 80) yang isi datanya *"GET /index.php?ask=kat&idkategori=3%27 HTTP/1.1"*. Pada paket data terdapat karakter %27, karakter ini merupakan simbol tanda petik.

b. Mencari Jumlah Kolom

Pada proses ini dilakukan dengan menambahkan *query* pada url dengan *syntax order by*, untuk menemukan jumlah kolom pada suatu tabel maka dilakukan dengan mengetikkan pada *browser* dengan url http://192.168.168.3/index.php?ask=kat&idkategori=3+order+by+1--, lakukan hal ini terus menerus dengan menaikkan nilai satu sampai muncul *error* seperti Gambar 3.16.



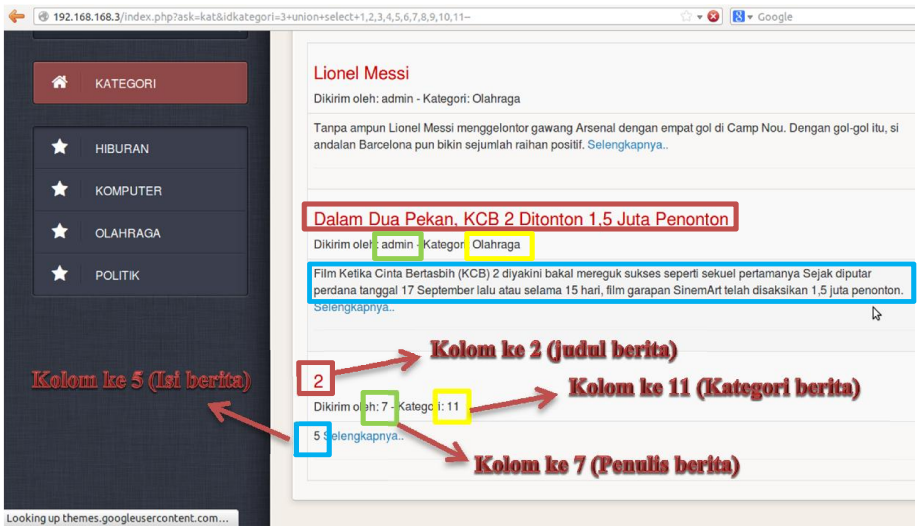
Gambar 3.16 Error MySQL Order Clause

Dapat dilihat pada Gambar 3.16 terdapat *error*, maka dapat disimpulkan bahwa jumlah kolom pada tabel yang ditampilkan pada *page* ini yaitu berjumlah sebelas kolom.

c. Menemukan Informasi *Database*

Setelah menemukan jumlah kolom yang ditampilkan maka selanjutnya menemukan kolom berapa yang digunakan untuk diambil datanya dan ditampilkan pada *page* tersebut. Dalam proses ini dilakukan dengan mengetikkan url http://192.168.168.3/index.php?ask=kat&idkategori=3+union+select+1,2,3,4,5,6,7,8,9,10,11--. Fungsi *union* pada hal ini adalah untuk

menggabungkan *record* dengan nilai 1,2,3,4,5,6,7,8,9,10,11 pada kolom 1,2,3,4,5,6,7,8,9,10,11.



Gambar 3.17 Informasi Kolom yang Digunakan

Pada Gambar 3.17 dapat diketahui kolom yang diambil datanya adalah kolom 2,7,11 dan 5. Dengan celah ini dapat diketahui nama *database* dan versi dari MySQL *server* dengan menggunakan perintah database() dan version(). Informasi mengenai versi mysql bertujuan untuk melihat kelemahan/*bug* dari versi tersebut. Untuk versi mysql > 5 celah ditemukan pada *database* informati_schema. Proses untuk menemukan database dan versi adalah dengan menggunakan kolom yang ditemukan dengan mengisi kolom tersebut dengan version() dan database(). Proses dilakukan dengan menggunakan url http://192.168.168.3/index.php?ask=kat&idkategori=3+union+select+1,database(),3,4,version(),6,7,8,9,10,11--, pada url ini digunakan kolom kedua dan kelima sebagai kolom yang akan menampilkan nama *database* dan versinya.



Gambar 3.18 Menemukan Database dan Versi MySQL

d. Menemukan Nama *Table*

Untuk menemukan *record-record* penting pada sebuah *database* perlu diketahui dahulu *table-table* pada *database* tersebut. Salah satu cara yang dilakukan *attacker* adalah dengan menggunakan url http://192.168.168.3/index.php?ask=kat&idkategori=-3+union+select+1,group_concat(table_name),3,4,5,6,7,8,9,10,11+from+information_schema.tables+where+table_schema=database() --. Dengan mengeksekusi url tersebut *attacker* dapat melihat *table-table* yang ada di *database* dbberita.

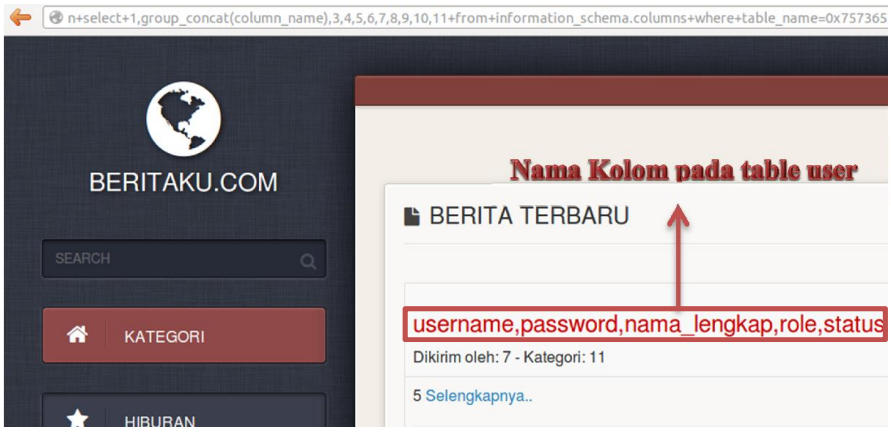


Gambar 3.19 Informasi Table pada Database dbberita

Pada Gambar 3.19 terlihat semua *table* yaitu tberita, tkategori, tkomentar, trating, tsubkategori, dan user. Dari semua *table* tersebut dapat dipastikan informasi penting berada pada *table* user. Maka langkah selanjutnya *attacker* akan mencari *record-record* yang terdapat pada *table* user.

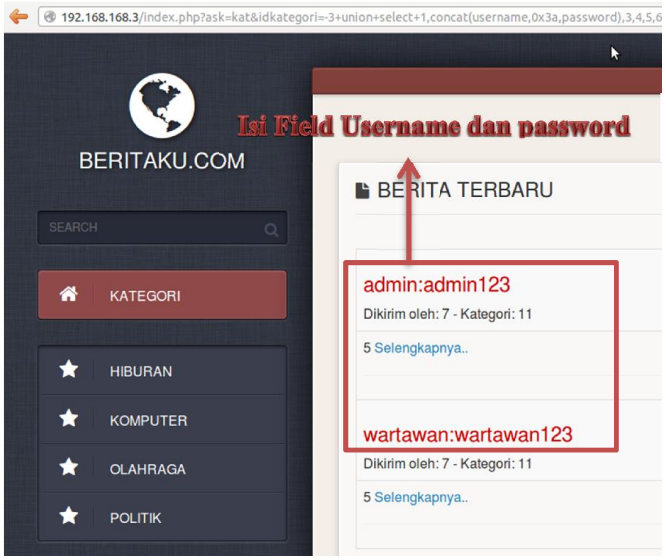
e. Menemukan *Record-Record* Penting

Pada tahap ini *attacker* sudah mendapatkan nama *table* yang akan dilihat *record-record*-nya. Pada kasus ini *attacker* akan melihat *table* user, maka *attacker* akan menggunakan salah satu cara yaitu dengan mengetikkan url http://192.168.168.3/index.php?ask=kat&idkategori=3+union+select+1,group_concat(column_name),3,4,5,6,7,8,9,10,11+from+information_schema.columns+where+table_name=0x75736572 --. Pada url terdapat table_name = 0x75736572, bilangan hexadesimal itu adalah string dari "user". Dengan mengeksekusi url tersebut *attacker* akan mendapatkan nama kolom pada *table* user, hasil dari eksekusi url tersebut dapat dilihat pada Gambar 3.20.



Gambar 3.20 Infomasi Nama Kolom Pada Database

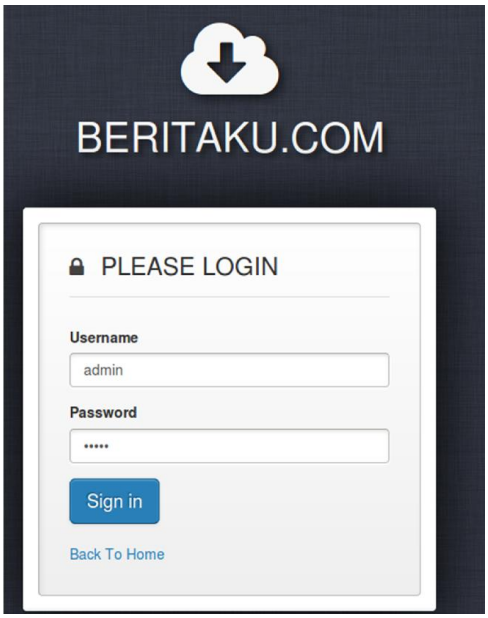
Pada Gambar 3.20 telah ditemukan nama kolom pada *table* user yaitu username, password, nama_lengkap, role, status. Langkah yang dilakukan *attacker* selanjutnya adalah menemukan *record-record* pada setiap *field*. Maka *attacker* menggunakan salah satu cara dengan mengeksekusi url http://192.168.168.3/index.php?ask=kat&idkategori=3+union+select+1,concat(username,0x3a,password),3,4,5,6,7,8,9,10,11+from+user--. Pada url tersebut *attacker* akan menampilkan *record* yang ada pada *field* username dan password. Hasil eksekusi dapat dilihat pada Gambar 3.21.



Gambar 3.21 Informasi *Record* Username dan Password

f. Mencari *Page* Admin

Dalam proses ini *attacker* sudah mendapatkan username dan password, langkah selanjutnya adalah menemukan *page* admin/login berada. Untuk kasus ini *page* admin berada pada http://192.168.168.3/admin/. Langkah selanjutnya *attacker* dapat memasukkan *username* dan *password* yang telah didapatkan pada proses sebelumnya.



Gambar 3.22 Page Login

Setelah melakukan *Sign In* maka *atacker* sudah dapat masuk ke dalam sistem, di dalam sistem *attacker* dapat menambahkan, menghapus, dan mengedit berita, kategori, dan user pada *web* tersebut.



Gambar 3.23 Login Berhasil

g. Menemukan Celah Pada *Login Page*

Pada proses ini tidak *attacker* tidak perlu melakukan langkah pada poin a sampai dengan e, *attacker* hanya perlu menemukan *login page*. *Query* SQL pada proses login yang banyak digunakan programmer yaitu :

```
select * from user where username='$user' and password='$password'
```

Pada *query* di atas dapat di manipulasi oleh *attacker* dengan mengisikan pada isian $user **toni' or 1=1 limit 0,1 #** dan $password toni(tidak ditentukan). Maka dengan isian seperti diatas Query SQL menjadi :

```
select * from user where username=toni' or 1=1 limit 0,1 #' and
password='$password'
```

Query SQL diatas akan menyeleksi sebuah *record* pada *table* user yang urutannya berada paling atas. *Attacker* akan mencoba celah tersebut pada *login page* seperti pada Gambar 3.24. Ketika tombol sign in ditekan maka akan seperti pada Gambar 3.25.

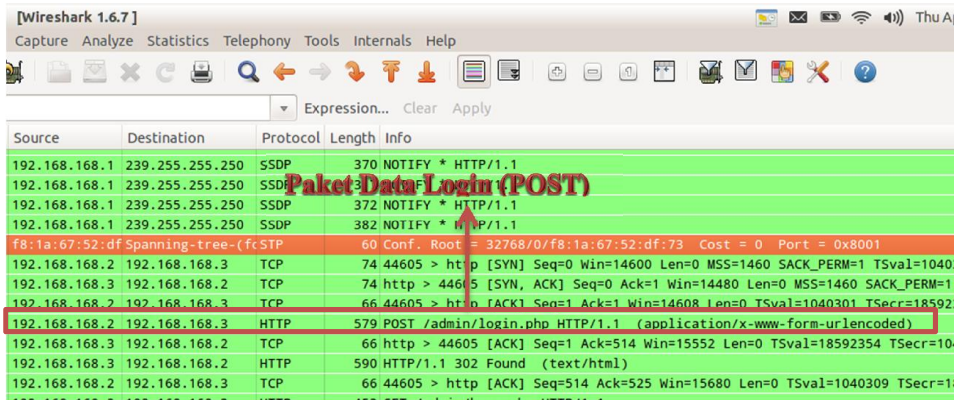


Gambar 3.24 Metode SQL *Injection* Pada *Login Page*



Gambar 3.25 Login Berhasil Menggunakan Metode SQL *injection*

Metode seperti diatas dapat dideteksi dengan menganalisa paket datanya, paket data dari proses ini yaitu :



Gambar 3.26 Analisa Paket Data SQL *Injection* Login Page

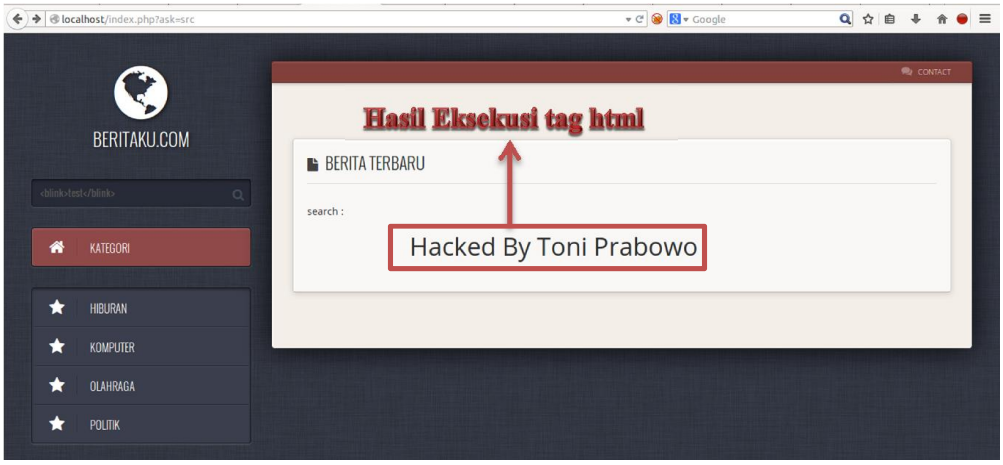
```
POST /admin/login.php  HTTP/1.1
Host:  192.168.168.3
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:28.0) Gecko/ 20100101 Firefox/28.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language:  en-US,en;q=0.5
Accept-Encoding : gzip, deflate
```

```
Referer: http://192.168.168.3/admin/
Cookie: PHPSESSID=d90j5nh t8ah0ube0fov3dk4 r67
Connection:  keep-alive
Content-Type: application/x-www-for m-urlencoded
Content-Length: 55
user=toni%27+or+1%3D1+limit+0%2C1+%23&pass=toni&submit=
```

Paket data dapat di deteksi dengan melihat karakter-karakter kutip(‘), 1=1, limit, #, dan lain-lain.

3.1.2.5 *Cross Site Scripting*

Cross Site Scripting adalah metode ancaman yang memaksa situs *web* untuk menampilkan kode berbahaya, yang kemudian dijalankan pada *browser web* pengguna. Metode penyerangan ini memanfaatkan celah pada kode program yang tidak di filter atau validasi pada sebuah inputannya. Penyerangan dengan metode ini dapat dilakukan dengan menyisipkan kode html pada kolom isian, banyak terjadi kasus XSS ini celah tersebut dimasukkan kode program untuk mendapatkan data-data dari pengguna website tersebut. Contoh penyerangan pada penelitian ini yaitu dengan memasukkan syntax html ke kolom search pada *web* dari komputer target “`<marquee> <h1>Hacked by Toni Prabowo</h1></marquee>`”. Hasilnya dapat dilihat pada Gambar 3.27.



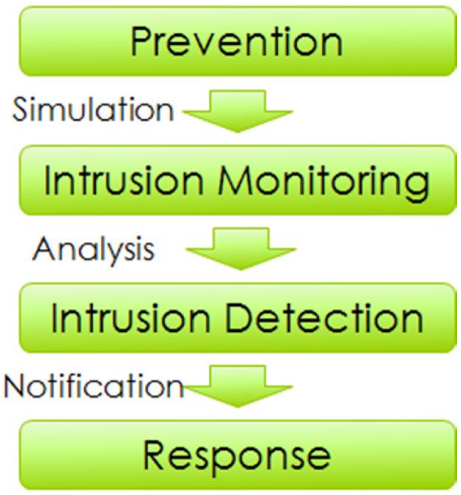
Gambar 3.27 Contoh Penyerangan XSS

3.1.3 Analisis Metode Deteksi Serangan

Metode pendeteksian serangan-serangan pada jaringan komputer dapat menggunakan salah satu sistem pendeteksian yaitu *intrusion detection system* (IDS). *Intrusion detection system* adalah sebagai *tool*, metode, sumber daya yang memberikan bantuan untuk melakukan identifikasi dan memberikan laporan terhadap aktifitas jaringan komputer. IDS mampu mencatat hasil pendeteksiannya ke dalam sebuah file atau database, sehingga administrator dapat melihatnya dikemudian hari.

3.1.3.1 Analisis *Intrusion Detection System*

Intrusion Detection System (IDS) dapat didefinisikan sebagai *tool*, metode atau sumber daya yang memberikan bantuan untuk melakukan identifikasi dan memberikan laporan terhadap aktivitas jaringan komputer. IDS merupakan bagian kecil dari sistem keamanan jaringan komputer.



Gambar 3.28 Aktivitas *Intrusion Detection System*

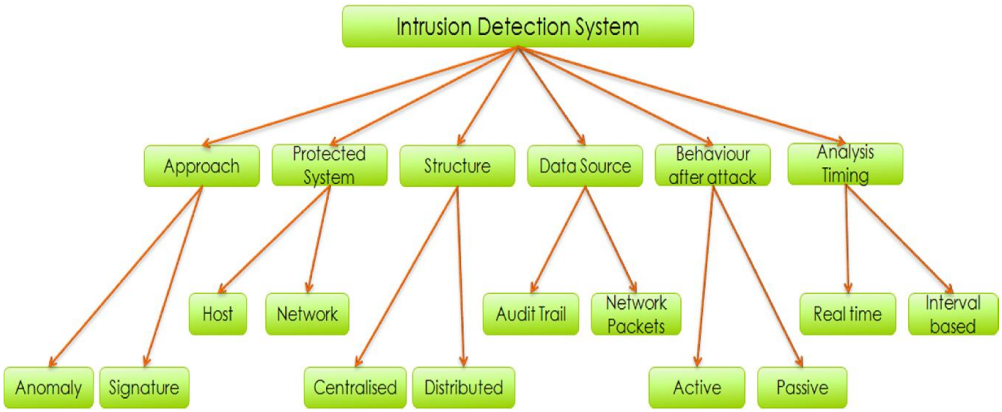
Tugas utama *dari intrusion detection system* adalah sistem keamanan komputer dengan mendeteksi serangan dan dapat dimungkinkan untuk mencatat serangan tersebut. Seperti yang digambarkan pada Gambar 3.28 terdapat bagian *prevention*, pada bagian ini *intrusion detection system* medeskripsikan jenis pencegahan yang akan dilakukan, jadi IDS perlu mengetahui jenis serangan apa

saja yang akan dicegah dengan menyimpan konfigurasi atau rule sebelumnya. Selanjutnya pada aktifitas *intrusion monitoring*, IDS melakukan *monitoring* terhadap paket-paket data yang ada di jaringan. Pada aktifitas Intrusion detection, IDS melakukan perbandingan paket-paket data yang ada pada jaringan dengan jenis pencegahan yang telah ditetapkan pada bagian *prevention*. Pada aktifitas respons, IDS melakukan respons baik itu *alert* atau *log* jika ada paket data yang sesuai dengan jenis serangan yang telah ditetapkan.

Ada beberapa alasan mengapa jaringan komputer perlu menggunakan IDS, diantaranya adalah :

1. Mendeteksi serangan dan pelanggaran keamanan sistem jaringan yang tidak bisa dideteksi oleh sistem yang umum digunakan seperti *firewall*.
2. Mendeteksi serangan awal. *Attacker* yang akan meyerang server biasanya melakukan langkah-langkah awal seperti *scan port*, menyimpan *backdoor*, dan lain-lain.
3. Menyediakan informasi yang akurat terhadap gangguan pada jaringan secara langsung, meningkatkan diagnosis, *recovery* dan mengkoreksi faktor-faktor penyebab serangan. Ketika IDS tidak mampu menghalangi suatu serangan, IDS akan mengumpulkan informasi dari peristiwa, serangan, dan jenis serangan yang terjadi, sehingga upaya perbaikan sistem jaringan menjadi lebih mudah dilakukan.

Intrusion detection system memiliki beberapa kategori dan klasifikasi seperti yang terdapat pada Gambar 3.29.

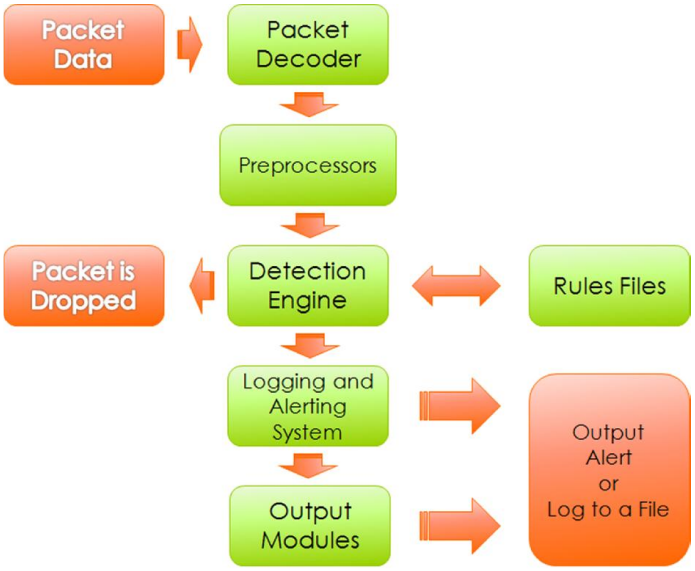


Gambar 3.29 Klasifikasi kategori *intrusion detection system*

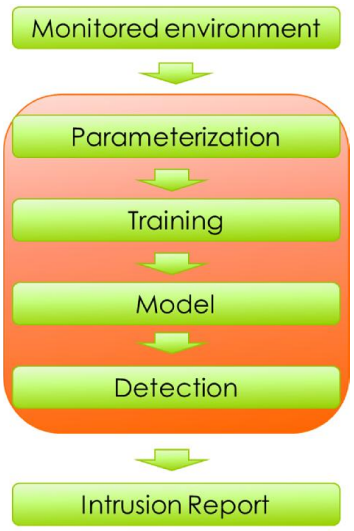
Dari kategori yang digambarkan pada Gambar 3.29 dapat dijelaskan bahwa IDS memiliki beberapa klasifikasi cara atau metode dalam penerapannya untuk mendeteksi serangan, berikut ini akan dijelaskan beberapa keuntungan dan kerugian dari jenis-jenis klasifikasinya.

3.1.3.1.1 *Anomaly vs. Signature*

Pada dasarnya terdapat dua pendekatan atau metode pada *intrusion detection system* yaitu *anomaly based* dan *signature based*. Masing-masing metode memiliki teknik yang berbeda dalam pendeteksian serangan pada suatu jaringan. Metode *signature based* bekerja seperti *antivirus*, yaitu dengan mendeteksi *patern-patern intrusion* yang ada pada *rules* atau *database* yang terdapat pada IDS tersebut. *Software* IDS akan mengumpulkan paket data yang selanjutnya akan di bandingkan dengan *rules* yang sudah didefinisikan dan selanjutnya akan memutuskan apakah paket data tersebut serangan atau bukan (Lihat Gambar 3.30). Kelemahan metode ini yaitu tidak dapat mendeteksi serangan yang baru apabila rule atau *database* tidak di *update*. Sedangkan metode *anomaly based* adalah metode pendeteksian serangan pada IDS dengan cara memonitor pola-pola pada jaringan, jika terdapat pola paket data yang mencurigakan maka akan dianggap sebagai sebuah interusi.Kelemahan metode ini yaitu banyak menghasilkan peringatan interusi yang sebenarnya paket data itu bukan merupakan serangan (*false alarm*). Proses yang dilakukan oleh metode *anomaly* dapat dilihat pada Gambar 3.31.



Gambar 3.30 Proses *Signature Based Detection*



Gambar 3.31 Proses *Anomaly Detection*

Dari kedua metode tersebut terdapat beberapa keunggulan dan kelemahan dari masing-masing metode, berikut akan dijelaskan perbandingan keunggulan diantara kedua metode tersebut pada Tabel 3.6.

Tabel 3.6 Perbandingan *Anomaly* dan *Signature based*

No	Anomaly based	Signature based
1	Memungkinkan untuk mendeteksi jenis serangan yang baru.	Mendeteksi serangan yang sudah dibuatkan rule-nya.
2	Banyak menghasilkan kesalahan pada saat mendeteksi serangan (*false alarm*).	Peringatan terjadinya interusi akan muncul jika sesuai dengan rule yang ditetapkan.
3	Paket data tidak akan dipantau selama proses pembangunan pelatihan profile serangan.	Tidak ada proses pelatihan dan pembangunan profile serangan.

Dari hasil penelitan yang dilakukan oleh Noviana Sagita dalam penelitianya yang berjudul "*Perbandingan Performansi antara Signature Based dan Anomaly Based dalam Pendeteksian Intrusi*" pada tahun 2011, telah dilakukan beberapa pengujian terhadap kedua metode ini dengan melakukan beberapa penyerangan dan hasilnya dapat dilihat pada Tabel 3.7.

Tabel 3.7 Hasil Pengujian Serangan terhadap metode IDS

Jenis Pengujian	Anomali (ourmon)	Signature (snort)
Port Scanning	Tidak Terdeteksi	Terdeteksi
SYN Flooding attack	Terdeteksi	Terdeteksi
Downloading File	Terdeteksi	Tidak Terdeteksi

Kesimpulan dari pengujian itu yaitu metode *signature* dapat mendeteksi dua pengujian dan tidak dapat mendeteksi pengujian *downloading file*, sedangkan metode *anomaly* dapat mendeteksi dua pengujian dan tidak dapat mendeteksi penguian *port scanning*. Pengujian *downloading file* bukan merupakan jenis aktifitas serangan. Contoh aplikasi yang menggunakan metode anomaly yaitu AAFID, MIDAS, NADIR, UNICORN, dan lain-lain. Contoh aplikasi IDS yang menggunakan metode signature yaitu ARMD, ASIM, Bro, CSM, CyberCop, GRIDS, Snort, Stalker, Suricata, Tripwire dan lain-lain.

3.1.3.1.2 *Host vs. Network*

Interusion detection system dapat digunakan untuk mendeteksi paket data yang tidak normal dengan ditempatkan pada dua tempat yaitu pada host dan network. Maka ada dua jenis proteksi system yang ada pada IDS yaitu *host intrusion detection system* (HIDS) dan *network intrusion detection system* (NIDS).

1. *Host Intrusion Detection System* (HIDS)

   IDS ini umumnya digunakan untuk kemanan-kemanan yang berhubungan dengan file pada suatu *host* atau *server*. Misalnya ada atau tidaknya *file* yang telah diubah data-datanya atau usaha untuk mendapatkan akses ke *file-file* yang berada pada *server*.

   HIDS memperoleh informasi dari dari data yang dihasilkan oleh sistem pada sebuah komputer yang diamati. Data HIDS biasanya berupa *log* yang dihasilkan dengan memonitor sistem *file*, *event*, dan kemanan pada *windows* dan *syslog* pada lingkungan sistem operasi UNIX. Saat terjadi perubahan pada *log* tersebut maka dilakukan analisis apakah sama dengan pola serangan yang ada pada basis data IDS. Teknik yang sering digunakan pada HIDS adalah dengan melakukan pengecekan pada *file* dengan *checksum* pada interval waktu tertentu untuk mendapatkan informasi mengenai perubahan suatu *file*. Kelemahan dari HIDS ini yaitu :

   a. Informasi harus dikonfigurasi untuk setiap host yang ada.
   b. Tidak cocok digunakan untuk melakukan monitoring paket data jaringan.
   c. *Host based* dapat dilumpuhkan dengan serangan DoS

   Contoh aplikasi HIDS yaitu ARMD, MIDAS, dan Tripwire.

2. *Network Intrusion Detection System* (NIDS)

   IDS dengan tipe ini umum digunakan untuk memonitoring paket-paket data pada sebuah jaringan. IDS ini mengumpulkan paket-paket data tersebut dan kemudian menganalisanya serta menentukan apakah paket itu berupa suatu paket yang normal atau suatu serangan atau berupa aktivitas yang mencurigakan. NIDS mendapatkan informasi dari paket-paket jaringan yang

ada. NIDS menggunakan *raw packet* yang ada di jaringan sebagai sumber datanya. Kelemahan dari NIDS ini yaitu :

a. Tidak dapat menganalisa paket yang dienkripsi.
b. Tidak dapat mengetahui apakah suatu serangan telah berhasil menghancurkan sistem, NIDS hanya mampu memberitahukan bahwa serangan sedang terjadi.
c. Tidak dapat memonitor perubahan data yang terjadi pada *server*.

Contoh aplikasi pada NIDS yaitu ASIM, Bro, CSM, CyberCop, GRIDS, Snort.

Dapat disimpulkan perbedaan-perbedaan antara HIDS dan NIDS pada Tabel 3.8.

Tabel 3.8 Perbedaan NIDS dan HIDS

No	NIDS	HIDS
1	Ruang lingkup yang luas (mengamati semua aktifitas jaringan)	Ruang lingkup yang terbatas (mengamati hanya pada host tertentu)
2	Lebih mudah melakukan setup	Setup yang kompleks
3	Leih baik untuk mendeteksi serangan yang berasal dari luar jaringan	Lebih baik mendeteksi serangan yang berasal dari dalam jaringan
4	Menganalisa paket header	Paket header diabaikan
5	Response *real time*	Response tergantung konfigurasi timing
6	Sistem Operasi independen	Sistem Operasi harus spesifik

Selain perbedaan, ada beberapa perbandingan dari HIDS dan NIDS, perbandingan ini diambil dari hasil analisa yang dilakukan oleh Ricky M. Magalhaes yang ditulisnya dalam artikel yang berjudul "*Host-Based IDS vs Network-Based IDS*" pada website www.windowsecurity.com. Pada Tabel 3.9 dipaparkan beberapa analisis antara HIDS dan NIDS.

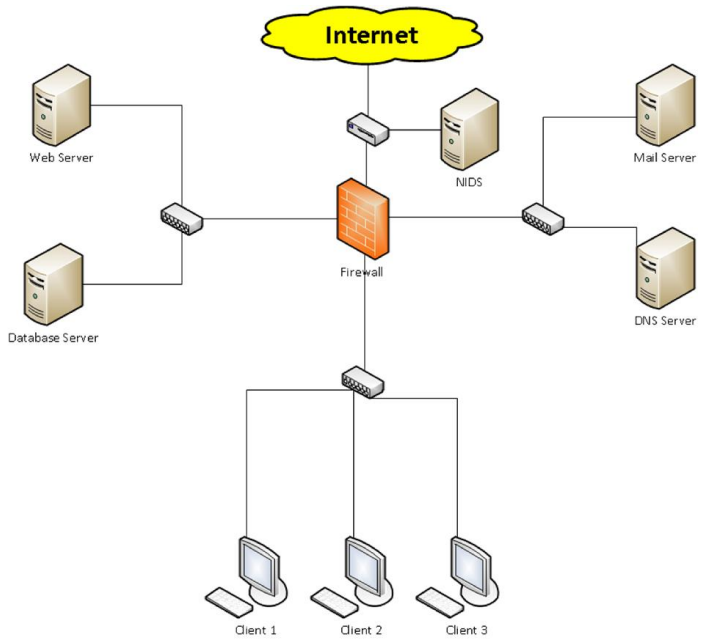
Tabel 3.9 Analisis Perbandingan antara HIDS dan NIDS

No	Function	HIDS	NIDS
1	Protection on LAN	****	****
2	Protection off LAN	****	-
3	Ease of Administration	****	****
4	Versatility	****	**
5	Price	***	*
6	Ease of Implementation	****	**
7	Little Training required	****	**
8	Total cost of ownership	***	**
9	Bandwidth requirements on (LAN)	0	2
10	Network overhead	1	2
11	Bandwidth requirements (internet)	**	**
12	Spanning port switching requirements	-	****
13	Update frequency to clients	****	-
14	Cross platform compatibility	**	****
15	Local machine registry scans	****	-
16	Logging	***	***
17	Alarm functions	***	***
18	PAN scan	****	-
19	Packet rejection	-	****
20	Specialist knowledge	***	****
21	Central management	**	***
22	Disable risk factor	*	****

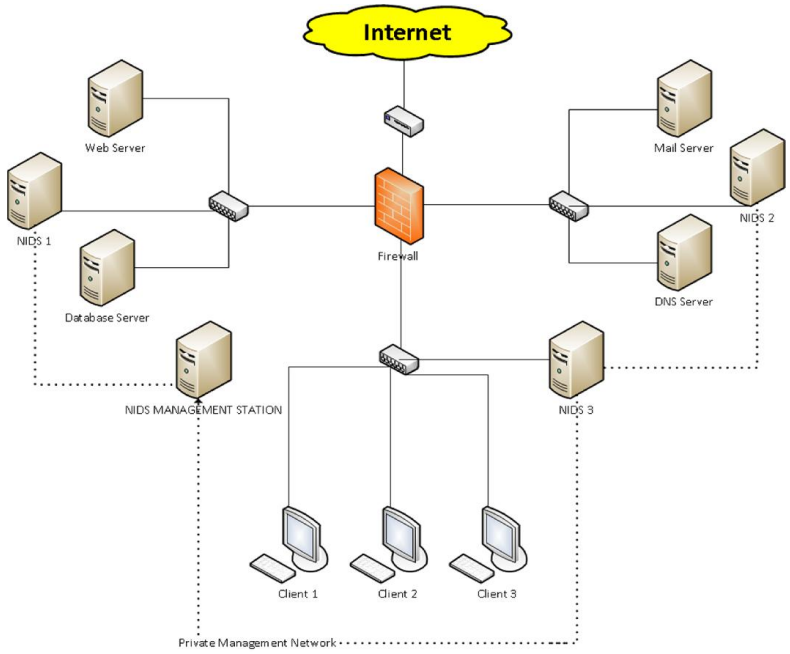
No	Function	HIDS	NIDS
23	Upgrade potential	***	***
24	Multiple LAN detection nodes	****	**

3.1.3.1.3 *Centralised vs. Distributed*

Intrusion detection system memiliki dua tipe infrastruktur yaitu *centralised instrusion detection system* (CIDS) dan *distributed intrusion detection system* (DIDS). CIDS yaitu tipe IDS yang seluruh kendalinya baik itu monitoring, deteksi dan pelaporannya dikendalikan secara terpusat. Infrastruktur CIDS dapat dilihat pada Gambar 3.32. Sedangkan DIDS adalah tipe IDS yang monitoring dan pendeteksian dilakukan oleh beberapa node lokal dengan pelaporan pada satu atau beberapa pusat. Infrastruktur untuk jenis DIDS dapat di lihat pada Gambar 3.33.



Gambar 3.32 Infrastruktur *Centralised Intrusion Detection*



Gambar 3.33 Infrastruktur *Distributed Intrusion Detection*

Model CIDS dan DIDS memiliki beberapa kelemahan dan keunggulan masing-masing terlihat pada

Tabel 3.10 Perbandingan CIDS dan DIDS

No	Centralised	Distributed
1	Mudah dalam melakukan konfigurasi.	Relatif sulit dalam konfigurasi
2	Biaya yang dikeluarkan cukup murah karena hanya menggunakan satu IDS.	Biaya yang dikeluarkan relatif mahal karena menggunakan beberapa IDS.
3	Tidak dapat memonitor paket data jika trafik sedang padat.	Dapat memonitor paket data hingga 1000 Mbs tergantung dari jumlah NIDS nya.

Dari Tabel 3.10 didapat kesimpulan bahwa CIDS lebih cocok digunakan pada jaringan komputer yang memiliki *host* sedikit, contoh aplikasi yang menggunakan CIDS adalah ARMD, ASIM, Bro, CMDS, CSM, CyberCop, MIDAS, NADIR, NIDAS, Stalker, Tripwire, dan lain-lain. Sedangkan untuk jaringan yang memiliki *host* banyak disarankan menggunakan DIDS, contoh

aplikasi IDS yang menggunakan DIDS yaitu AAFID, CSM, GRIDS, Snort, dan lain-lain.

3.1.3.1.4 *Audit Trail vs. Network Packets*

Intrusion detection system memiliki dua tipe pengambilan sumber data yaitu *audit trail* dan *network packets*. Audit trail biasanya digunakan oleh jenis HIDS, IDS akan mengenali suatu serangan dari *file log* yang ada pada *server*, sedangkan untuk *network packets,* IDS akan mengenali serangan dari paket-paket data pada jaringan. Dari kedua jenis sumber data yang digunakan IDS ini memiliki beberapa kelemahan dan keunggulan, diantaranya dijelaskan pada Tabel 3.11.

Tabel 3.11 Perbandingan *Audit Trail* dan *Network Packet*

No	Audit Trail	Network Packet
1	Monitoring data tidak *real-time.*	Monitoring data *real-time*.
2	Lebih lambat dalam proses pendeteksian.	Cepat dalam mengenali suatu serangan.
3	Diterapkan pada HIDS	Diterapkan pada NIDS

Contoh aplikasi IDS yang menggunakan *audit trail* yaitu ASIM, NADIR, Stalker, Tripwire. Contoh aplikasi IDS yang menggunakan *network packet* yaitu AAFID, ARMD, Bro, CMDS, CSM. CyberCop, MIDAS, Snort, dan lain-lain.

3.1.3.1.5 *Active vs. Passive*

Intrusion detection system memiliki dua aksi yang diterapkan jika mendeteksi suatu pola serangan yaitu active dan passive. IDS active adalah IDS yang akan melakukan aksi berupa pemblokiran terhadap host yang diduga melakukan penyerangan biasanya IDS jenis ini digabungkan dengan IPS (*intrusion Prevention System*) berupa *firewall* atau *proxy server*. Sedangkan IDS passive yaitu IDS yang hanya melakukan pencatatan file dan pelaporan kepada administrator jika terdeteksi pola serangan.

3.1.3.1.6 *Real time vs. Interval-based*

Intrusion detection system memiliki dua jenis analisis *timing* yaitu *real time* dan *interval-based.* IDS realtime adalah IDS yang melakukan pendeteksian paket data secara *real time*, setiap data yang lewat pada jaringan akan di analisa paket datanya. IDS *interval based* adalah IDS yang melakukan pendeteksian berdasarkan waktu yang ditetapkan, IDS jenis ini akan meng-*audit file log* pada *host* untuk menganalisa paket datanya.

3.1.3.2 Analisis Pemilihan Intrusion Detection System

Pemilihan IDS yang baik adalah dengan menganalisa jaringan yang sudah ada dan beberapa serangan yang dimungkinkan terjadi pada jaringan tersebut. Pada latar belakang masalah penelitian ini jenis serangan-serangan yang akan deteksi adalah serangan yang memungkinkan untuk menyerang web server, sehingga beberapa faktor penunjang dalam pemilihan IDS yaitu seperti pada Tabel 3.12.

Tabel 3.12 Pemilihan Jenis IDS

No	Klasifikasi IDS	Jenis Klasifikasi Yang Dipilih	Alasan
1	Approach	Signature based	Karena dari hasil analisis penyerangan pada point 3.1.2 didapat kesimpulan bahwa metode-metode penyerangan pada web server seperti SQL injection dan XSS tidak dapat menggunakan metode anomaly based. (Lihat Poin 3.1.3.1.1)
2	Protected System	Network	Karena dapat memper-mudah pada saat penerap-an. Web Server tidak perlu di-*install* IDS didalamnya, *web server* hanya perlu di-daftarkan IP addressnya pada Server IDS.

No	Klasifikasi IDS	Jenis Klasifikasi Yang Dipilih	Alasan
3	Structure	Centralised	Karena scope jaringan yang dimonitor paketnya hanya pada jaringan lokal pada penempatan web server berada, sehingga jumlah traffik paket data pada jaringan tersebut tidak tinggi. (Lihat Poin 3.1.3.1.3)
4	Data Source	Network Packets	Karena IDS ini diletakkan pada sebuah server khusus yang terhubung dengan mirror port pada switch sehingga sumber data yang dianalisa bersumber dari jaringan. (Lihat Poin 3.1.3.1.4)
5	Behaviour after attack	Passive	Karena IDS yang diperlukan tidak akan melakukan aksi untuk memblok akses dari pengguna yang mengakses web, IDS yang diperlukan hanya sebagai pencatat apabila terjadi serangan dan melaporkannya kepada administrator bahwa web server dengan ip address sekian memiliki *vuln.* (Lihat Poin 3.1.3.1.5)
6	Analisis Timing	Real Time	Karena data bersumber dari jaringan sehingga setiap paket data yang lewat akan langsung di analisa dan dibandingkan dengan parameter yang telah ditetapkan. (Lihat Poin 3.1.3.1.6)

Selain pemilihan IDS berdasarkan klasifikasi pada Tabel 3.12 ada satu faktor dalam pemilihan IDS yaitu license, karena IDS yang akan digunakan hanya

sebagai bahan penelitian maka pemilihan IDS menggunakan license *open source* agar proses dari IDS dapat dianalisa lebih detail.

Setelah mengklasifikasikan IDS yang dipilih, maka selanjutnya adalah menentukan aplikasi yang akan digunakan sebagai IDS, pemilihan dilakukan dengan membandingkan beberapa aplikasi IDS yang berlicense *open source*. Dari analisa berbagai sumber maka ada delapan aplikasi IDS open source yang akan dibandingkan. Perbandingan aplikasi IDS dapat dilihat pada Tabel 3.13.

Tabel 3.13 Perbandingan Aplikasi IDS Open Source

No	Aplikasi	Klasifikasi IDS					
		1	2	3	4	5	6
1	Bro	A & S	N	C & D	N	A / P	R
2	Kismet	S	N	D	N	P	R
3	Open DLP	S	H	D	A	P	I
4	Ossec	S	H	D	A	P	R
5	Samhain	S	H	C	A	P	I
6	Snort	S	N	C & D	N	A / P	R
7	Suricata	S	N	C & D	N	A / P	R
8	Tripwire	S	H	C	A	P	I

Keterangan :

1 : Approach	A=Anomaly	S=Signature
2 : Protected System	H=Host	N=Network
3 : Structure	C=Centralised	D=Distributed
4 : Data Source	A=Audit Trail	N=Network Packet
5 : Behaviour After Attack	A=Active	P=Passive
6 : Analisys Timing	R=Real Time	I=Interval-based

Dari hasil perbandingan pada Tabel 3.13 aplikasi yang memenuhi kriteria yaitu Bro, Snort dan Suricata. Dari ketiga aplikasi tersebut dibandingkan kembali dari fitur-fitur yang ada, diambil dari hasil penelitian yang dilakukan oleh Mauno Pihelgas yang berjudul "*A Comparative Analysis Of Open-Source Intrusion Detection Systems*" dalam penelitian tersebut dilakukan perbandingan aplikasi IDS yaitu Bro, Suricata dan Snort. Hasil perbandingan fitur-fitur ketiga aplikasi tersebut dapat dilihat pada Tabel 3.14.

Tabel 3.14 Perbandingan Fitur dari Bro, Suricata dan Snort

No	Parameter	Snort	Suricata	Bro
1	IPS Feature	Snort_inline or snort used with -Q option	optional while compiling (--enable- nfqueue)	No
2	Rules	VRT::Snort rules SO rules Emerging Threats rules	VRT::Snort rules EmergingThreats rules	Pre-packaged scripts
3	Threads	Single-thread	Multi-thread	Single-thread
4	Installation complexity	Relatively straightforward. Installation also available from packages.	Relatively straightforward. Not available from packages (except Ubuntu).	Relatively straightforward. Installation also available from packages.
5	Documentation	Well documented on the official website and many resources on the Internet.	Well documented on the official website. Some resources on the Internet.	Satisfactory documentation on the official website. Very few resources on the Internet
6	Event logging	Flat file, database, unified2 logs		Flat file, data-base,barnyard2 integration

No	Parameter	Snort	Suricata	Bro
7	IPv6 support	Yes, when compiled with --enable-ipv6 option.	Yes	Yes
8	Capture Accelerators	Yes (e.g. PF_RING)		
9	Offline analysis (pcap file)	Yes, multiple files consecutively	Yes, only single file	Yes, only single file
10	Frontends	Sguil, Aanval, BASE, FPCGUI (Full Packet Capture GUI), Snortsnarf		Brownian
11	License	GNU GPL v.2		BSD

Dari perbandingan pada Tabel 3.14 maka pada penelitian ini aplikasi yang akan digunakan sebagai IDS yaitu snort. Karena aplikasi snort sudah memiliki banyak *documentation* di dunia, snort banyak digunakan oleh orang-orang sebagai bahan penelitian juga.

3.1.3.3 Analisis *Intrusion Detection System* Snort

Snort merupakan aplikasi keamanan yang digunakan untuk mendeteksi interusi-interusi jaringan (penyusupan, penyerangan, pemindaian, dan beragam bentuk ancaman lainnya). Snort mampu menganalisis dan membentuk *logging* paket-paket yang melintasi jaringan secara *real time* berbasis TCP/IP. Metode yang digunakan snort dalam mengenali intrusi adalah metode *signature based*. Paket jaringan akan dikenali sebagai serangan jika paket data tersebut sesuai dengan pola serangan yang dimiliki oleh snort atau di sebut juga *rules*. Metode *signature based* pada snort menggunakan algoritma *Boyer-Moore (string matching)*.

Snort dapat dikonfigurasikan untuk berjalan pada mode-mode berikut :

1. *Sniffer Mode*

   Digunakan untuk melihat paket-paket dari jaringan dalam bentuk teks pada *console* (layar).

2. *Packet Logger Mode*

   Digunakan untuk mencatat paket-paket data pada jaringan ke dalam *disk* untuk dianalisis dikemudian hari.

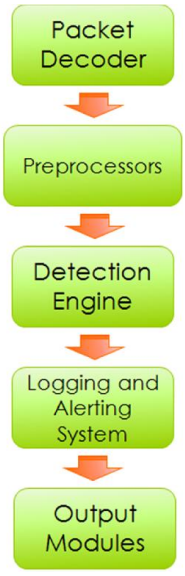
3. *Network Intrusion Detection System* (NIDS) *Mode*

   Digunakan untuk mendeteksi pola serangan pada jaringan yang sesuai dengan *rules* yang telah dibuat.

4. *Inline Mode*

   Digunakan untuk mengambil paket dari *iptables* dan mengintruksikan *iptables* untuk menolak atau meneruskan paket tersebut berdasarkan jenis rule snort yang digunakan [9].

Snort memiliki enam komponen dasar yang bekerja saling berhubungan, komponen-komponennya yaitu *Packet Decoder*, *Preprocessors*, *Rules Files*, *Detection Engine*, *Logging* and *Alerting* System, dan *Output Modules*.



Gambar 3.34 Komponen Snort

Pada aplikasi snort terdapat beberapa perintah-perintah untuk menjalankan aplikasinya. Perintah-perintah tersebut dapat dilihat pada Tabel 3.15.

Tabel 3.15 Perintah-perintah pada Snort

Perintah	Fungsi
-A	Set Alert (fast, full, console, none)
-b	Log packet in tcpdump format
-c <rules>	Use Rules Files <rules>
-C	Print out payloads with character data only
-d	Dump the application Layer
-e	Display the second layer headder info
-f	Turn off fflush() calls after binary log writes
-F <bpf>	Read BPF filters from file
-G <0xid>	Log Identifier
-h <hn>	Home network
-i <if>	Listen on interface
-I	Add interface name to alert output
-k <mode>	Checksum mode (all,noip,notcp,noudp,noicmp,none>
-K <mode>	Logging mode (pcap[default],ascii,none)
-l <ld>	Log to directory
-L <file>	Log to this tcpdump file
-n <cnt>	Exit after receiving <cnt> pakets
-N	Turn off logging (alerts still work)
-o	Change the rule testing order to Pass|Alert|Log
-O	Obfuscate the logged IP addresses

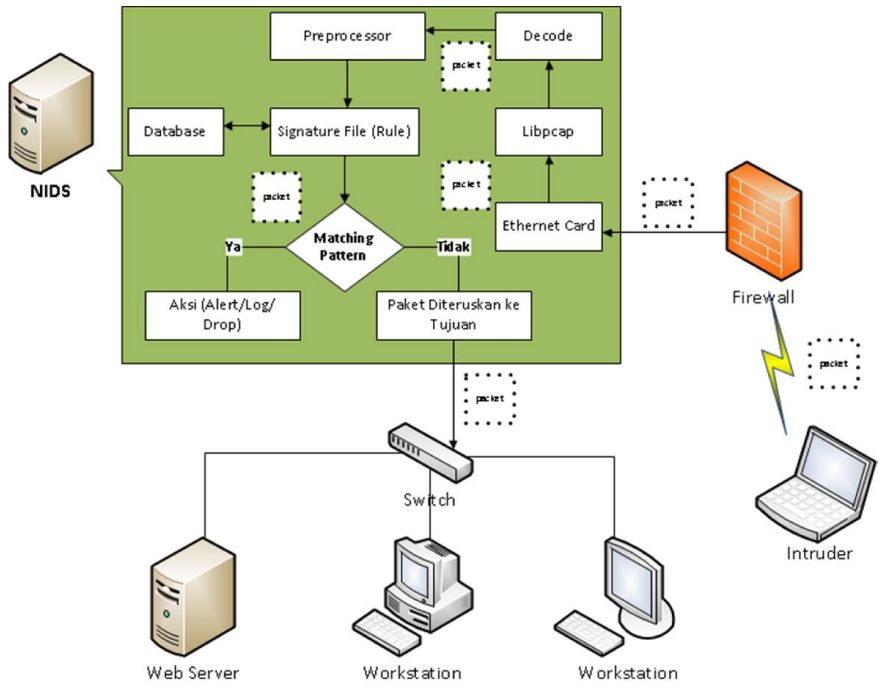
Perintah	Fungsi
-p	Disable promiscuous mode sniffing
-P <snap>	Set explicit snaplen of packet (default: 1514)
-q	Quiet. Don't show banner and status report
-r <tf>	Read and process tcpdump file <tf>
-R <id>	Include 'id' in snort_intf<id>.pid file name
-s	Log alert messages to syslog
-S <n=v>	Set rules file variable n equal to value v
-T	Test and report on the current snort configuration
-U	Use UTC for timestamps
-v	Be verbose
-V	Show version number
-w	Dump 802.11 management and control frames
-X	Dump the raw packet data starting at link layer
-y	Include year in timestamp in the alert and log files
-Z	Set the performonitor preprocessor file path and name
-z	Set assurance mode, match on established sessions (for TCP)

Contoh penggunaan untuk menjalankan aplikasi snort sebagai *sniffing* paket data dengan menampilkan paket data berupa hex dan teks, maka perintah yang diketikkan yaitu `snort -v -d -e`.

3.1.3.4 Analisis Proses Pada Snort

Pada jaringan komputer terdapat beberapa perangkat yang melakukan komunikasi data, data-data tersebut dikirim dan diterima dari satu perangkat ke perangkat lainnya. Paket data tersebut ada yang normal dan tidak normal dengan kata lain ada paket data yang berupa ancaman yang dapat menyusup ke dalam

sebuah sistem komputer. Maka untuk mendeteksi paket data yang normal dan tidak normal dibutuhkan IDS untuk mendeteksinya, dalam penelitian ini digunakan aplikasi IDS snort. Proses analisa paket data pada aplikasi snort dapat dilihat pada Gambar 3.35.



Gambar 3.35 Proses Snort Mendeteksi Paket Data

Pada Gambar 3.35 digambarkan proses analisis paket data yang dilakukan oleh snort dimulai dengan meng-*capture* paket data menggunakan modul libpcap, lalu data tersebut di-*decode*, selanjutnya masuk pada tahap *preprocessor*, selanjutnya paket data dibandingkan dengan *rule* yang sudah ada, jika paket data cocok dengan dengan *rule* yang ada maka dilakukan aksi sedangkan jika tidak maka paket data akan dilanjutkan ke *switch*. Untuk lebih jelas maka akan dilakukan penjelasan lebih rinci bagaimana fungsi masing-masing modul tersebut.

3.1.3.4.1 Analisis Pcap

Pcap (*packet capture*) adalah sebuah application programming interface (API) untuk meng-*capture* traffik jaringan komputer. Pada sistem operasi unix dikenal dengan libpcap dan pada sistem operasi windows dikenal dengan nama

winpcap. Pcap dikembangkan menggunakan bahasa pemrograman C. API pcap banyak digunakan oleh beberapa aplikasi monitoring jaringan. Pcap akan meng-capture trafik dan menyimpannya pada sebuah file dengan eksetensi *.cap. Format data yang digunakan akan sama apabila pcap ini dijalankan pada sistem operasi unix atau windows. Sebuah file cap memiliki *global header* yang mengandung beberapa *global information* yang diikuti oleh banyak *record* untuk setiap paket yang di-*capture*, seperti yang diperlihatkan pada Gambar 3.36.



Gambar 3.36 Struktur *File* Pcap

Struktur data pada global *header* yaitu :

```
typedef struct pcap_hdr_s {
  guint32 magic_number;
  guint16 version_major;
  guint16 version_minor;
  gint32  thiszone;
  guint32 sigfigs;
  guint32 snaplen;
  guint32 network;
} pcap_hdr_t;
```

Penjelasan dari masing-masing variable dapat dilihat pada Tabel 3.16.

Tabel 3.16 Penjelasan Variable Global Header

No	Variable	Fungsi
1	`magic_number`	Digunakan untuk menyimpan format file.
2	`version_major` dan `version_minor`	Digunakan untuk menyimpan jenis versi file.
3	`thiszone`	Digunakan untuk menyimpan lokal time zone.

No	Variable	Fungsi
4	sigfigs	Digunakan untuk menyimpan tingkat akurasi waktu yang ditulus, pada prakteknya variable ini biasa diisi 0
5	snaplen	Digunakan untuk menyimpan nilai maksimal paket yang dapat di-capture.
6	network	Digunakan untuk menyimpan tipe link-layer. Seperti 802.11, PPP, Token Ring, FDDI, dan lain-lain.

Struktur data pada *packet header* yaitu :

```
typedef struct pcaprec_hdr_s {
  guint32 ts_sec;
  guint32 ts_usec;
  guint32 incl_len;
  guint32 orig_len;
} pcaprec_hdr_t;
```

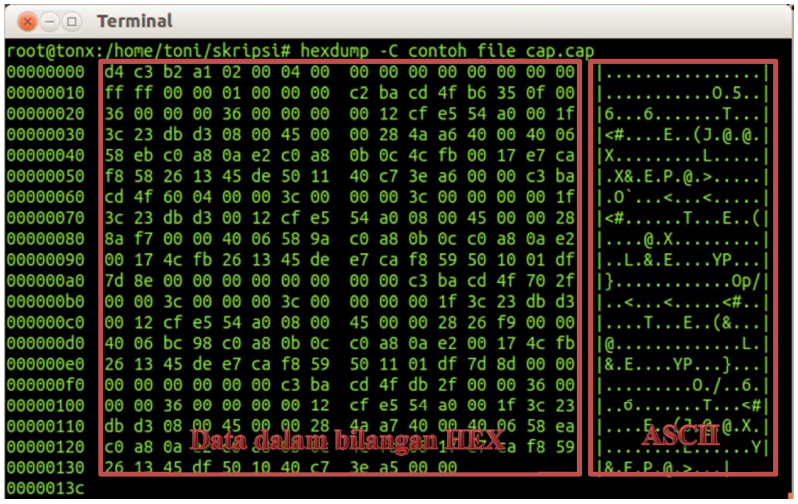
Penjelasan dari masing-masing variable dapat dilihat pada Tabel 3.16.

Tabel 3.17 Penjelasan Variable Packet Header

No	Variable	Fungsi
1	ts_sec	Digunakan untuk menyimpan waktu (date and time) pada saat paket data di-*capture*.
2	ts_usec	Digunakan untuk menyimpan waktu dalam format microseconds.
3	incl_len	Digunakan untuk menyimpan jumlah paket data yang di-*capture* dan disimpan pada file.
4	orig_len	Digunakan untuk menyimpan panjang paket yang muncul pada jaringan ketika paket data tersebut di-*capture*.

Untuk paket data tidak ada struktur data, isi dari packet data ini merupakan data yang di-*capture* pada saat paket muncul pada jaringan. Untuk lebih jelasnya

berikut pada Gambar 3.37 merupakan contoh *file* Pcap yang dibuka menggunakan aplikasi hexdump di linux.



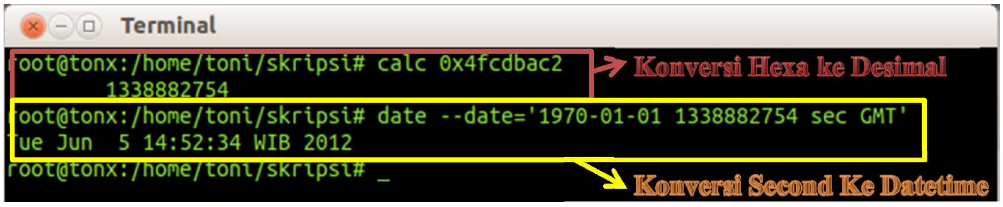
Gambar 3.37 Contoh File Pcap

Dari contoh *file* pcap pada Gambar 3.37 maka dapat dilihat bagian-bagian dari *global header*, *packet header* dan *packet data*. Berikut ini dijelaskan pada Tabel 3.18 isi dari data tersebut.

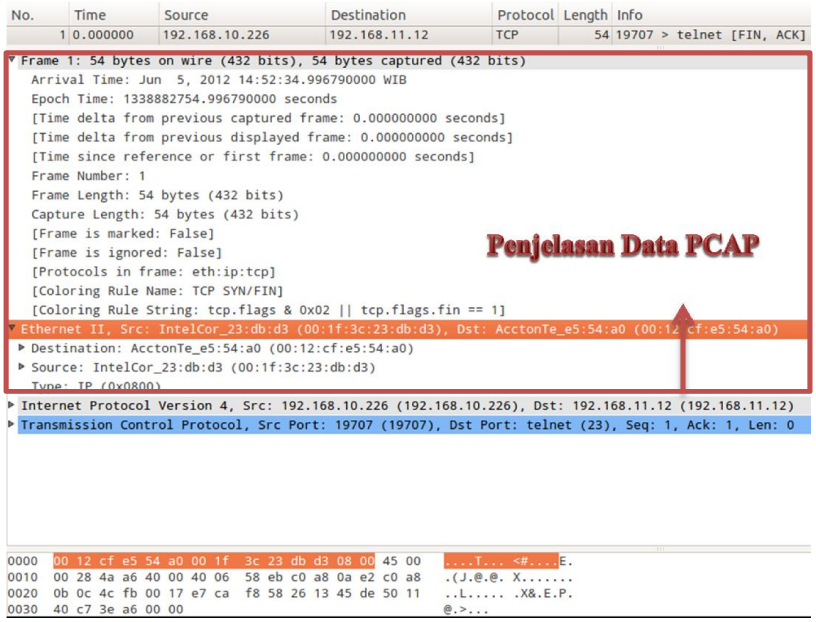
Tabel 3.18 Penjelasan Data Dari File Pcap

No	Data	Penjelasan
1	d4 c3 b2 a1	Magic number (4 bytes) 0xd4c3b2a1
2	02 00 04 00	Major version (2 bytes) dan minor version (2 bytes) jika didesimalkan maka versinya adalah 2.4
3	00 00 00 00 00 00 00 00	Ini merupakan GMT dan tingkat akurasi waktu.
4	ff ff 00 00	Ini merupakan nilai maksimal dari paket data, 0xffff = 65535 bytes
5	01 00 00 00	Ini adalah tipe link-layer yang digunakan 0x1 merupakan ethernet.

No	Data	Penjelasan
6	c2 ba cd 4f	Ini merupakan timestamps dalam satuan detik. Jika di desimalkan maka 0x4fcdbac2 = 1338882754 second terhitung dari 1970-01-01. Jika dihitung maka hasilnya selasa, 5 juni 14:52:34 WIB 2012. (Lihat Gambar 3.38)
7	b6 35 0f 00	Ini merupakan waktu pada saat paket data di capture. 0x000f35b6 = 996790 microseconds
8	36 00 00 00	Ini merupakan nilai dari panjang paket yang yang ada di dalam file
9	36 00 00 00	Ini merupakan nilai dari panjang paket data yang di-capture
10	00 12 s.d. a6 00 00	Ini merupakan paket data yang berisi mac address, ip address, dll. Lihat Gambar 3.39.
11	c3 ba dan seterusnya	Merupakan isi dari paket header dan paket data seperti yang telah dijelaskan pada no 6 s.d 10.



Gambar 3.38 Perhitungan Konversi Waktu



Gambar 3.39 Isi dari Paket Data Pada File Pcap

3.1.3.4.2 Analisis *Decoder*

Setelah libpcap meng-*capture* paket data yang ada pada jaringan, selanjutnya paket data tersebut akang di-*decode* oleh modul/komponen *decoder*. Komponen ini digunakan untuk men-*decode* paket data dalam bentuk struktur data yang nantinya akan digunakan oleh *preprocessor*. Struktur data tersebut meliputi identifikasi protokol, *decode* IP, *port number*, *IP address*, dan lain-lain. Komponen ini juga akan memberikan *alert* jika menemukan suatu paket yang cacat[5]. Contoh paket data (lanjutan dari paket data pada poin 3.1.3.4.1):

```
00 12 cf e5 54 a0 00 1f 3c 23 db d3 08 00 45 00 00 28
4a a6 40 00 40 06 58 eb c0 a8 0a e2 c0 a8 0b 0c 4c fb
00 17 e7 ca f8 58 26 13 45 de 50 11 40 c7 3e a6 00 00
```

Paket data ini akan dipisah-pisahkan menurut jenisnya, pada paket data tersebut dapat terlihat paket data tersebut terdiri dari MAC *address source* dan *destination*, *IP address*. Untuk lebih jelasnya dijelaskan pada Tabel 3.19.

Tabel 3.19 Penjelasan *Packet Decoder*

Data	Penjelasan
00 12 cf e5 54 a0	Mac address destination 00:12:cf:e5:54:a0
00 1f 3c 23 db d3	Mac address Source (*Attacker*) 00:1f:3c:23:db:d3
08 00	Header IP
45 00	Header Length 20 bytes
00 28	Total Length 40 bytes
4a a6	Identification 0x4aa6 (19110)
40 00	Fragment Offset : 0
40	Time to live : 64
06	Protocol TCP
58 eb	Checksum
c0 a8 0a e2	IP address source 192.168.10.226
c0 a8 0b 0c	IP address destination 192.168.11.12
4c fb 00 17 e7 ca f8 58 26 13 45 de 50 11 40 c7 3e a6 00 00	Port source 4c fb: 19707 Port destination 00 17 : 23 (Telnet) Seq : 1 Ack number : 1 Len:0 Flag : 0x011 (FIN, ACK)

Maka dengan proses ini snort sudah dapat mengetahui paket data tersebut menggunakan protokol TCP, *ip address* dan *port* dari *source* dan *destination* juga sudah diketahui, maka selanjutnya akan dilakukan proses *preprocessing*.

3.1.3.4.3 Analisis *Preprocessors*

Komponen ini digunakan snort untuk memodifikasi atau mengatur paket sebelum masuk pada komponen *detection engine*. *Preprocessor* juga dapat

mengenali paket yang aneh yang berada di paket *header* lalu akan mengirimkan peringatan. *Preprocessor* sangat penting pada IDS karena komponen ini menyiapkan paket data yang akan di analisis terhadap *rules* pada *detection engine*. Kode *preprocessor* dijalankan sebelum *detection engine* diaktifkan. Paket data dapat dimodifikasi atau dianalisis menggunakan mekanisme ini. *Preprocessor di-load* dan dikonfigurasi menggunakan *keyword preprocessor*. Format untuk mengarahkan `preprocessor` dalam *file rule* pada snort adalah `preprocessor <name>: <options>.`

Preprocessor pada snort memiliki *rules* seperti pada komponen *detection engine*. Rule pada *preprocessor* digunakan untuk mendeteksi paket yang cacat dan dapat juga mendeteksi serangan dari jenis protokol paket data tersebut. Contoh pendeteksian yang dilakukan oleh *preprocessor* adalah deteksi *port scanning*. *Rules* untuk *preprocessor* ini biasanya disimpan pada *folder* `/etc/snort/preproc_rules`. Jenis-jenis *preprocessor* yang dapat dikonfigurasi pada aplikasi snort sesuai dengan snort *manual* dokumen terdapat 22 jenis *preprocessor* yang dapat dikonfigurasi. Preprocessor tersebut dijelaskan pada Tabel 3.20.

Tabel 3.20 Preprocessor Snort

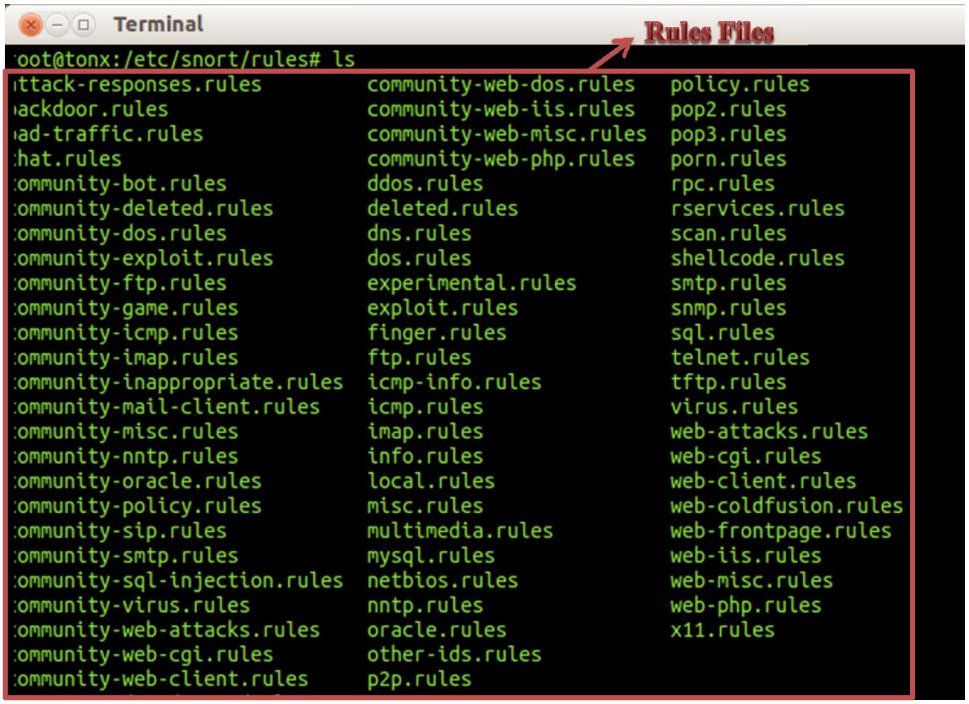
No	Preprocessor	Fungsi
1	Frag3	Tujuan dari preprocessor ini yaitu : 1. Ekesekusi lebih cepat dengan manajemen data yang tidak serumit frag2. 2. Teknik-teknik anti-pengelakan berbasis target host modeling.
2	Stream5	Digunakan untuk melacak sessions pada protokol TCP dan UDP
3	sfPortscan	Digunakan untuk mendeteksi pengintaian yang dilakukan oleh attacker
4	RPC Decode	Digunakan untuk menormalkan beberapa RPC yang terfragmentasi ke dalam sebuah record yang tidak terfragmentasi.

No	Preprocessor	Fungsi
5	Performance Monitor	Digunakan untuk menampilkan statistik dari pengukuran snort.
6	HTTP Inspect	Bertanggung jawab untuk men-decode-kan traffik http dan mendeteksi lapisan aplikasi serangan eksploit http design.[5]
7	SMTP Preprocessor	Digunakan untuk men-decode SMTP untuk aplikasi pengguna dan mencari perintah dan respons dari SMTP
8	POP Preprocessor	Digunakan untuk men-decode POP untuk aplikasi pengguna dan mencari perintah dan respons dari POP
9	IMAP Preprocessor	Digunakan untuk men-decode IMAP untuk aplikasi pengguna dan mencari perintah dan respons dari IMAP4
10	FTP/Telnet Preprocessor	Digunakan untuk men-decode FTP dan Telnet untuk aplikasi pengguna dan mencari perintah dan respons dari FTP dan Telnet
11	SSH Preprocessor	Digunakan untuk mendeteksi eksploitasi contohnya : hallenge-Response Buffer Overflow, CRC 32, Secure CRT, and the Protocol Mismatch exploit.
12	DNS Preprocessor	Digunakan untuk men-decode DNS respons dan mendeteksi eksploitasi contohnya : DNS Client RData Overflow, Obsolete Record Types, and Experimental Record Types
13	SSL/TLS Preprocessor	Digunakan untuk men-decode SSL dan TLS traffik.
14	ARP Spoof Preprocessor	Digunakan untuk men-decode paket ARP dan mendeteksi ARP attacks, unicast ARP request, dan inconsistent Ethernet to IP mapping
15	DCE/RPC 2 Preprocessor	Digunakan untuk SMB desegmentation dan DCE/RPC defragmentation

No	Preprocessor	Fungsi
16	Sensitive Data Preprocessor	Digunakan untuk mendeteksi dan menyaring Personally Identifiable Information (PII). Informasi ini termasuk nomor kartu kredit, U.S. Social Security numbers dan alamat email.
17	Normalizer	Digunakan untuk menormalkan paket yang bertujuan untuk memperkecil *the chances of evasion*
18	SIP Preprocessor	Digunakan untuk menyediakan cara untuk mengatasi vilnerabilities dasar dan Exposures yang terkait dengan SIP (*Session Initiation Protocol*).
19	Reputation Preprocessor	Digunakan untuk mem-blacklis/whitelist IP, untuk block/drop/pass traffik dari IP yang didaftarkan.
20	GTP Decoder and Preprocessor	Digunakan untuk mendeteksi intrusi yang melalui GTP (*GPRS Tunneling Protocol*)
21	Modbus Preprocessor	Digunakan untuk men-decode protokol modbus
22	DNP3 Preprocessor	Digunakan untuk men-decode protokol DNP3

3.1.3.4.4 Analisis *Rules Files*

Komponen ini merupakan suatu file teks yang berisi daftar aturan yang sintaksnya sudah diketahui. Sintaks ini meliputi protokol, *address*, *output plug-in* dan hal-hal yang berhubungan dengan berbagai hal. Pada sistem operasi *linux rule* biasanya berada pada direktori `/etc/snort/rules`. Berikut ini *rule-rule* yang sudah disediakan oleh snort version 2.9.2.



Gambar 3.40 Rules Snort Versi 2.9.2

Rules file dapat dimodifikasi, dihapus dan ditambah agar snort dapat mendeteksi serangan-serangan baru. *Rules files* juga dapat di-*download* di forum snort.org. *File rule* pada snort memiliki ekstensi *file* *.rules. Pada Gambar 3.40 belum ada *rules* untuk mendeteksi sql *injection* untuk MySQL dan deteksi serangan XSS. Maka pada penelitian ini akan di jelaskan bagaimana membuat rule untuk mendeteksi SQL *injection* pada *database* mySQL dan serangan *Cross Site Scripting* (XSS).

Rule-rule pada snort terbagi atas dua bagian utama :

1. *Rule Header*

   Merupakan bagian *rule* dimana aksi-aksi *rule* diidentifikasi. Aksi-aksi tersebut yaitu *alert*, *log*, *pass*, *active*, *dynamic*, dan lain-lain.

Tabel 3.21 Rule Header

Rule Header	Fungsi
`alert`	Membuat sebuah pesan peringatan
`log`	Instruksi untuk menyimpan paket
`pass`	Mengabaikan paket
`active`	Memberi pesan alert lalu mengaktifkan rule dinamis lainnya
`dynamic`	Tetap dalam status menunggu sampai diaktifkan oleh sebuah rule pengaktifan, lalu bertindak sebagai rule log
`drop`	Mengintruksikan iptable untuk membuang paket dan mencatatkan log untuknya.
`reject`	Menginstruksikan iptable untuk membuang paket lalu mengirimkan TCP reset jika protokolnya berupa TCP, atau sebuah pesan ICMP port unreachable jika protokolnya berupa UDP
`sdrop`	Mengintruksikan iptables untuk membuang paket tanpa menuliskan log

2. *Rule Options*

   Merupakan bagian *rule* dimana pesan-pesan peringatan diidentifikasi.[12]

Penulisan rule pada snort memiliki format khusus, contoh di bawah ini merupakan salah satu rule pada snort.

```
Alert tcp any any -> 192.168.168.3 21 (content:
"anonymous"; msg: "Ada yang coba FTP server!";)
```

Rule diatas digunakan untuk mendeteksi apabila ada yang mencoba mengakses FTP (port 21) dengan account yang belum terdaftar (*anonymous*) lalu akan mengirimkan alert dengan pesan "Ada yang coba FTP server!". Rule diatas memiliki format seperti yang dijelaskan pada Tabel 3.22.

Tabel 3.22 Penjelasan Rule Deteksi FTP Server

Rules	Penjelasan
`Alert`	Rule header dengan aksi alert
`Tcp`	Jenis Protocol
`Any`	Source IP address
`Any`	Source Port Number
`->`	Directional Operator
`192.168.168.3`	Destination IP address
`21`	Destination Port Number
`Content:"anonymous"`	Rule Option Payload detection
`Msg:"Ada yang coba FTP server!"`	Pesan alert

3.1.3.4.5 Analisis *Detection Engine*

Komponen ini bertanggung jawab untuk mendeteksi jika terjadi aktifitas intrusi yang berada pada paket. Paket data akan dibandingkan dengan *rules-rules* yang ada pada direktori $RULE_PATH. Jika sebuah paket data cocok dengan rule yang ada, maka snort akan melakukan aksi sesuai dengan aksi *rule* tersebut. Pada *mode* NIDS beberapa paket mungkin tidak dapat dideteksi jika trafik pada jaringan sangat tinggi. Kecepatan deteksi pada *detection engine* dipengaruhi oleh beberapa faktor yaitu :

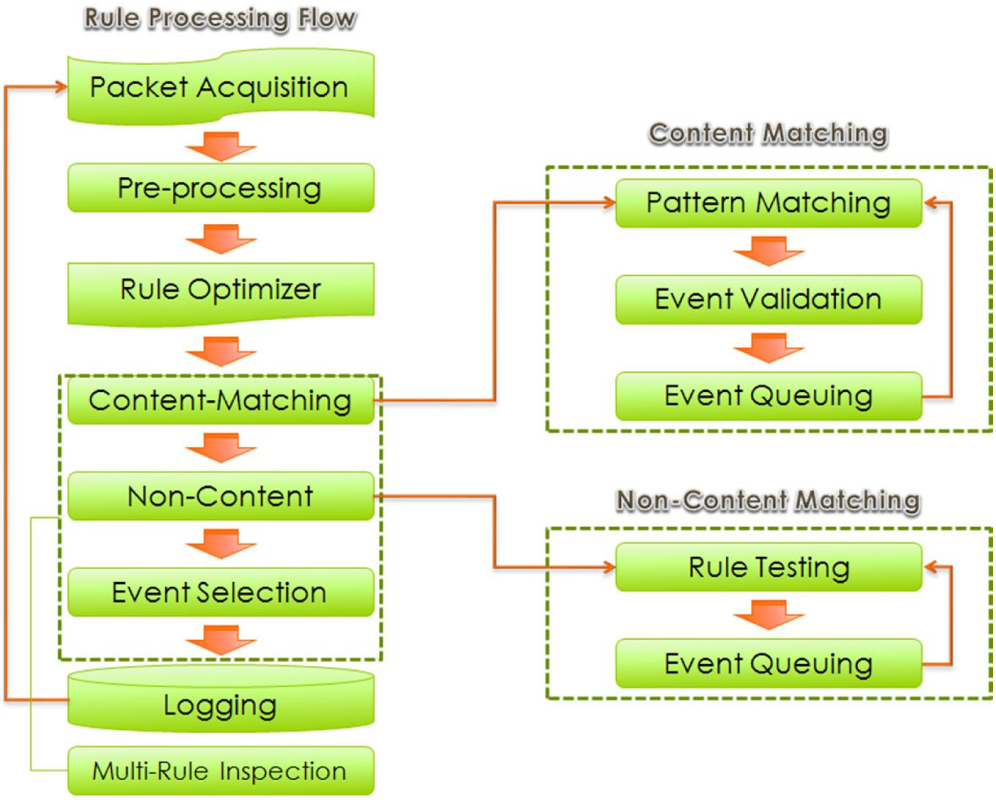
a. Banyaknya *rules* yang digunakan.
b. Spesifikasi *hardware* yang menjalankan snort.
c. Trafik data pada jaringan.

Pada saat mendeteksi sebuah paket data, *detection engine* akan membagi beberapa bagian dari paket tersebut. Bagian-bagiannya yaitu :

a. *IP Header* dari paket data
b. *Transport layer header*. Header ini meliputi TCP, UDP, ICMP dan lain-lain.

c. *Application layer level header*. Bagian ini meliputi DNS header, FTP header, SNMP header dan SMTP header.
d. *Packet payload*. Bagian ini merupakan data dari sebuah paket. Sehingga pada saat membuat rules dapat digunakan untuk mencari sebuah string yang ada pada paket tersebut.

Proses snort mengenali suatu paket secara garis besar adalah seperti pada Gambar 3.41.



Gambar 3.41 Proses Rule Mengenali Suatu Paket

Proses diawali dengan melakukan *preprocessing* dari paket-paket data yang berada sudah di *log*, proses *preprocessing* ini sudah dijelaskan pada poin 3.1.3.4.3, setelah melalui tahap *preprocessing* maka dilakukan *rule optimize* dilakukan untuk membuat sebuah rule menjadi lebih *efficient* dan membuat rule menjadi sedemikian rupa sehingga setiap paket data yang datang dapat dicari oleh sebuah rule yang ada. *Content matching* akan melakukan pengecekan *content*

yang ada pada setiap paket data. Setiap konten (*payload*) pada paket data dicocokan *pattern*-nya dengan rule yang ada pada snort. Jika paket data tidak memiliki *content* maka snort akan melakukan pencocokan pada rule-rule yang tidak memiliki konten juga.Selanjutnya dari proses pencocokan jika ditemukan paket data yang cocok dengan rule maka dilakukan aksi yang sudah ditentukan pada rule tersebut dan dilakukan penyimpanan pada *file log* snort.

Algoritma deteksi pada snort menggunakan algoritma Boyer-Moore *pattern matching*. Algoritma Boyer-Moore adalah salah satu algoritma untuk mencari *pattern* di dalam string yang dibuat oleh R.M Boyer dan J.S Moore. Tujuan utama algoritma ini adalah mencari *pattern* dengan melakukan perbandingan karakter dimulai dari karakter paling kanan dari *pattern* yang dicari. Snort akan mendeteksi isi *content* dengan menggunakan algoritma ini. *Keyword content* merupakan salah satu fitur snort yang cukup penting, *keyword* ini memungkinkan untuk mengeset rule yang mencari konten spesifik dalam *payload* paket dan memicu respons berdasarkan data tersebut. Kapan pun pencocokan sebuah pola opsi konten dibentuk maka fungsi pencocokan pola Boyer-Moore akan dipanggil. Jika data cocok dengan *string* yang ada di dalam *payload* paket maka akan dilakukan eksekusi selanjutnya. Fungsi ini bersifat *case sensitive*.

Algoritma Boyer-Moore merupakan algoritma pencocokan string yang ditemukan pada tahun 1977. Algoritma ini memiliki dua fase yaitu fase *preprocessing* dan fase pencarian. Algoritma Boyer-Moore menggunakan dua buah tabel yaitu tabel pertama yang akan menghitung banyaknya pergeseran yang harus dilakukan berdasarkan identitas karakter yang menyebabkan kegagalan pencocokan string tabel pertama ini disebut juga *bad character shift*, dan tabel kedua juga akan menghitung banyaknya pergeseran yang harus dilakukan, tetapi berdasarkan jumlah karakter yang berhasil dicocokan sebelum pencocokan string tersebut gagal. Pada penelitian ini akan dianalisa algoritma ini yang dicontohkan pada pembuatan program menggunakan bahasa C. Dalam kasus ini string yang akan di cari adalah “union” string ini adalah yang biasanya digunakan dalam serangan SQL *injection*, string tersebut akan dicari pada suatu paket data yang

memiliki data “http://192.168.168.3/index.php?ask=kat&idkategori=1+ union+ select+1,2,3,4,5,6,7,8,9,10,11--". Berikut ini adalah penjelasan yang algoritma Boyer-Moore.

Tahap pertama adalah menghitung nilai pergeseran dari string “union” yang dilakukan adalah mencari nilai *bad character shift* dan *good- suffix shift* dengan menggunakan fungsi berikut :

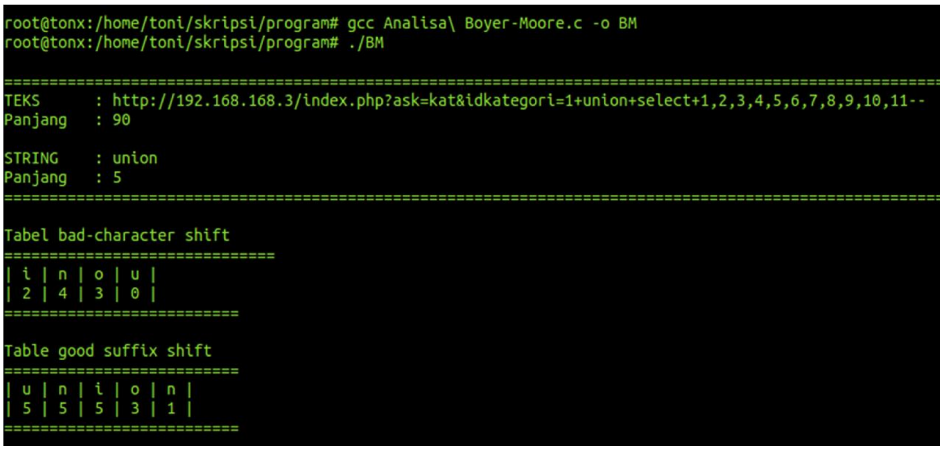
```
static void prepare_badcharacter_heuristic(const char
*str, size_t size,int result[ALPHABET_SIZE])
{
 size_t i;
 for (i = 0; i < ALPHABET_SIZE; i++)
   result[i] = -1;
 for (i = 0; i < size; i++)
   result[(size_t) str[i]] = i;
}


void prepare_goodsuffix_heuristic(const char *normal,
size_t size,int result[size + 1])
{
    char *left = (char *) normal;
    char *right = left + size;
    char reversed[size+1];
    char *tmp = reversed + size;
    size_t i;
    *tmp = 0;
    while (left < right)
     *(--tmp) = *(left++);
    int prefix_normal[size];
    int prefix_reversed[size];
    compute_prefix(normal, size, prefix_normal);
    compute_prefix(reversed, size, prefix_reversed);
    for (i = 0; i <= size; i++) {
```

```
            result[i] = size - prefix_normal[size-1];
        }

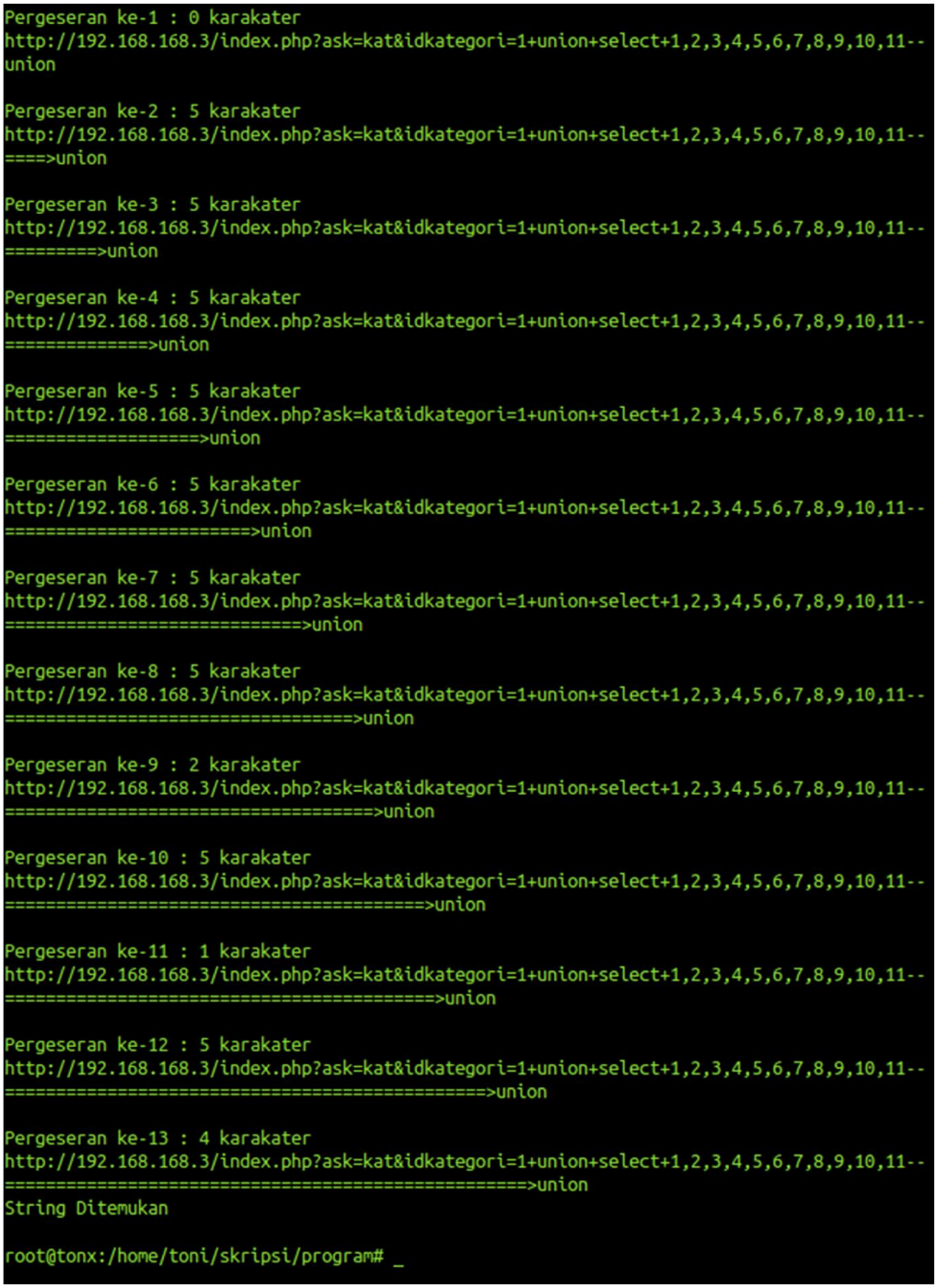
        for (i = 0; i < size; i++) {
            const int j = size - prefix_reversed[i];
            const int k = i - prefix_reversed[i]+1;
            if (result[j] > k) result[j] = k;
        }
}
```

Maka akan di dapat nilai pergeseran seperti pada Gambar 3.42.



Gambar 3.42 Proses Preprocessing Boyer-Moore

Tahapan selanjutnya adalah proses pencocokan string atau *pattern* pada teks yang akan dicari, dimana teks yang akan dicari adalah http://192.168.168.3/index.php?ask=kat&idkategori=1+union+select+1,2,3,4,5,6, 7,8,9,10,11--.



Gambar 3.43 Proses Pencarian Boyer-Moore

Proses pencarian ini dilakukan dengan proses membandingkan karakter paling kanan dari *pattern*, contoh pada Pergeseran ke 1 :

Tabel 3.23 Proses Pergeseran ke-1 Pencocokan Pattern

Kolom	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Teks	h	T	t	p	:	/	/	1	9	2	.	1	6	8	.
Pattern	u	n	i	o	n										

Pada Tabel 3.23 karakter ‘n’ dan ‘:’ tidak sama dan karakter ‘:’ tidak ada pada *pattern* “union” sehingga dilakukan pergeseran 5 karakater, maka proses selanjutnya adalah melakukan perbandingan karakter ‘n’ dan ‘2’, kedua karakter tersebut tidak sama dan karakter ‘2’ tidak ada pada *pattern* “union” maka lakukan pergeseran sebanyak 5 karakter. Tahap selanjutnya akan sama jika pencocokan karakter tidak sama dan karakter pada teks tidak ada pada pattern.

Tabel 3.24 Proses Pergeseran ke-2 Pencocokan Pattern

Kolom	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Teks	h	t	t	p	:	/	/	1	9	2	.	1	6	8	.
Pattern						u	n	i	o	n					

Untuk contoh pergeseran berbeda terlihat pada pergerseran ke-8. Pada pergeseran tersebut karakter ‘n’ dibandingkan dengan karakter ‘i’, kedua karakter tersebut tidak sama tetapi karakter ‘i’ ada pada pattern “union” sehingga nilai pergeseran mengacu pada tabel *bad character shift*, karakter ‘i’ memiliki nilai pergeseran 2, sehingga lakukan pergeseran selanjutnya sebanyak 2 karakter. Sehingga terlihat seperti pada Tabel 3.26.

Tabel 3.25 Proses Pergeseran ke-8 Pencocokan Pattern

Kolom	30	31	32	33	34	35	36	37	38	39	40	41	42	42	43
Teks	p	?	a	s	k	=	k	a	t	&	i	d	k	A	t
Pattern							u	n	i	o	n				

Tabel 3.26 Proses Pergeseran ke-9 Pencocokan Pattern

Kolom	30	31	32	33	34	35	36	37	38	39	40	41	42	42	43
Teks	p	?	a	s	k	=	k	a	t	&	i	d	k	A	t
Pattern									u	n	i	o	n		

Pada Tabel 3.26 setelah dilakukan pergeseran sebanyak 2 karakter lalu dibandingkan kembali karakter paling akhir dari *pattern* yaitu karakter ‘k’ dan ‘n’, kedua karakter tersebut tidak sama dan karakter ‘k’ tidak ada pada *pattern* “union” sehingga dilakukan pergeseran sebanyak 5 karakter. Selanjutnya pergeseran terakhir pada proses pergeseran ke -13, karakter ‘n’ dibandingkan dengan karakter ‘n’, karakter tersebut sama maka selanjutnya akan dilakukan perbandingan terhadap karakter ‘o’ (kolom ke-56) dengan karakter ‘o’ pada pattern, setelah dibandingkan ternyata karakter sama, maka dilakukan perbandingan pada karakter-karakter sebelumnya. Sehingga pada pergeseran ke-13 ditemukan *pattern* pada teks yang sama dengan *pattern* yang dicari, maka status pencarian adalah “ditemukan”. Pada snort apabila ditemukan *pattern* yang sesuai pada rule maka akan dilakukan aksi yang telah ditentukan pada rule tersebut.

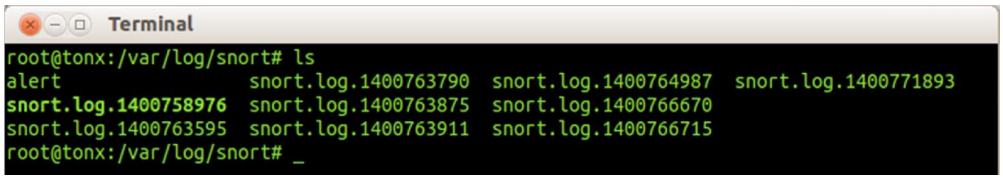
3.1.3.4.6 Analisis *Logging* and *Alerting System*

Komponen ini digunakan untuk proses logging dan alerting, tergantung dari rules yang dibuat apakah sebuah paket yang terdeteksi serangan itu paketnya akan disimpan atau disampaikan kepada administrator. Komponen ini merupakan tindakan yang akan dilakukan pada saat *log* atau *alert*. Bagaimana *log* atau *alert* itu dieksekusi. Contohnya yaitu:

a. *Log* disimpan di sebuah *file*
b. *Log* atau *alert* dikirim melalui SNMP
c. *Log* atau *alert* dikirim ke fasilitas syslog
d. *Logging* ke *database* seperti MySQL atau Oracle.

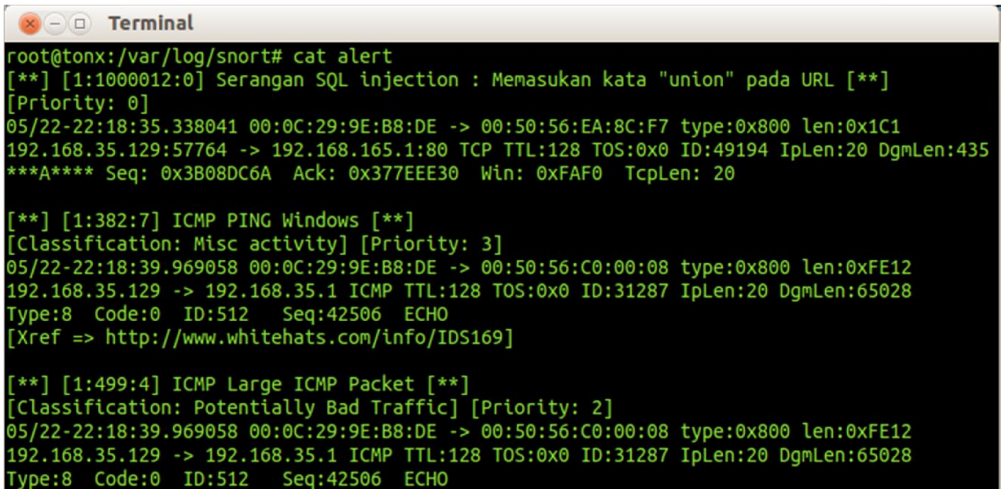
e. Membuat file XML.
f. Memodifikasi konfigurasi *router* dan *firewall*.
g. Mengirim *Server Message Block* (SMB) ke Microsoft *Windows-Based machine*.

Dengan menggunakan parameter –A full pada saat menjalankan aplikasi snort maka alert dan paket data akan di simpan dengan format file default yang biasanyanya disimpan pada directori `/var/log/snort/alert`. File Alert disimpan dengan filename `alert` sedangkan paket data log disimpan dengan filename `snort.log.<waktu dalam satuan detik>`.



Gambar 3.44 Direktori File Log Snort

Struktur data file alert yaitu seperti pada Gambar 3.45.



Gambar 3.45 Isi Data File Alert

Pada Gambar 3.45 terdapat beberapa informasi yang dijelaskan pada .Struktur data file snort.log.datetime yaitu memiliki format seperti file pcap.

Tabel 3.27 Penjelasan Data Pada File Alert Snort

No	Informasi	Penjelasan
1	[**] [1:382:7]	Informasi sid (id rule)
2	ICMP PING Windows [**]	Pesan Alert yang dihasilkan
3	[Classification: Misc activity] [Priority: 3]	Prioritas Alert
4	05/22-22:18:39.969058	Datetime alert
5	00:0C:29:9E:B8:DE -> 00:50:56:C0:00:08	Mac Address Sumber dan MacAddress Tujuan
6	type:0x800 len:0xFE12	Type TCP dengan panjang data 65042 bytes
7	192.168.35.129 -> 192.168.35.1	Ip Address Sumber dan Ip Address Tujuan
8	ICMP TTL:128 TOS:0x0 ID:31287 IpLen:20 DgmLen:65028 Type:8 Code:0 ID:512 Seq:42506 ECHO [Xref => http://www.whitehats.com/info/IDS169]	Informasi protokol, panjang data, dan beberapa informasi mengenai paket data yang dideteksi serangan.

3.1.4 Analisis Kebutuhan Infrastruktur

Kebutuhan infrastruktur terbagi menjadi dua yaitu kebutuhan perangkat keras dan kebutuhan perangkat lunak.

3.1.4.1 Kebutuhan Perangkat Keras

Perangkat Keras yang dibutuhkan pada penelitian ini dalam kasus simulai penyerangan dan pendeteksian terbagi menjadi dua kebutuhan yaitu perangkat keras pada *server* dan perangkat keras *attacker*.

a. Perangkat Keras *Server*

Server menyediakan layanan-layanan yang akan diakses oleh *client*, seperti *file server* dan *web server*. Untuk spesifikasinya adalah sebagai berikut:

Tabel 3.28 Spesifikasi Perangkat Keras *Server*

No	Komponen	Detail Spesifikasi
1	Processor	Intel Core i3
2	Memory	2 Gb DDR3
3	Harddisk	320 Gb
4	Operating System	Ubuntu 12.04
5	Network Interface Card	100 Mbps

b. Perangkat Keras *Attacker*

Komputer *attacker* digunakan untuk menyerang komputer *server*. Untuk spesifikasinya adalah sebagai berikut:

Tabel 3.29 Spesifikasi Perangkat Keras *Attacker*

No	Komponen	Detail Spesifikasi
1	Processor	Intel Dual Core
2	Memory	1 Gb DDR1
3	Harddisk	160 Gb
4	Operating System	Linux Debian
5	Network Interface Card	100 Mbps

3.1.4.2 Kebutuhan Perangkat Lunak

Perangkat lunak yang digunakan dalam penelitian ini adalah sebagai berikut :

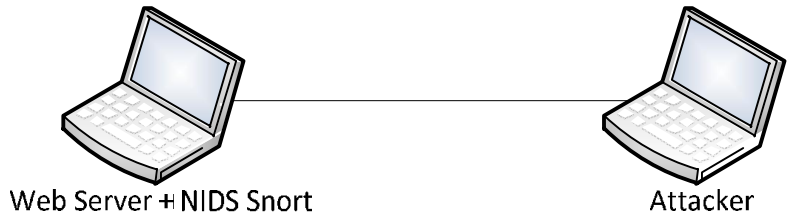
1. Sistem Operasi *web server* Linux Ubuntu 12.04
2. Sistem Operasi *attacker* menggunakan windows XP.
3. *Attacker* menggunakan aplikasi hping3.
4. *Attacker* menggunakan *browser* firefox.
5. *Network Intrusion Detection System* Snort 2.9.2.
6. *Database* untuk menyimpan data menggunakan MySQL 5.5.35.

3.2 Perancangan Sistem

Pada perancangan sistem akan dipaparkan mengenai perancangan arsitektur simulasi, perancangan arsitektur pengujian, perancangan *rule* pada snort, dan beberapa perancangan lainnya.

3.2.1 Perancangan Arsitektur Simulasi

Perancangan arsitektur jaringan yang digunakan untuk mensimulasikan bagaimana *web server* yang telah dipasang sebuah IDS mampu mendeteksi serangan-serangan oleh *attacker* terhadap *web* tersebut dengan *rules-rules* yang telah dibuat. Skema arsitektur simulasi tersebut ditunjukkan pada Gambar 3.46.

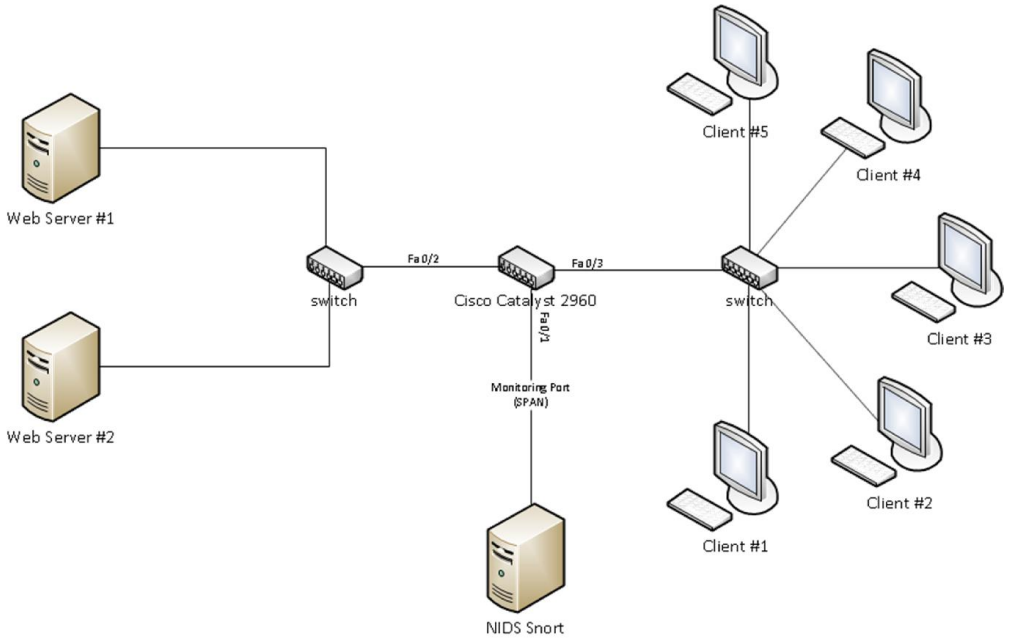


Gambar 3.46 Skema Jaringan untuk Simulasi Pendeteksian

Gambar 3.46 digunakan atau diterapkan pada saat pengujian penelitian ini kepada penguji sebagai bukti bahwa *rules* yang telah dibuat mampu mendeteksi serangan-serangan. Pada Gambar 3.46 *Attacker* menggunakan *operating system* windows xp yang berjalan pada vmware, sedangkan *web server* menggunakan *operating system* linux ubuntu 12.04.

3.2.2 Perancangan Arsitektur Pengujian

Perancangan arsitektur jaringan pada Gambar 3.47 digunakan untuk menguji IDS snort yang akan mendeteksi jenis-jenis serangan yang telah dibuat rule-rule nya. Pengujian akan dilakukan di Telkom Corporate University di gedung NITS lab CPE. Pada skema jaringan ini terdapat dua *web server* yang akan dijadikan target serangan dan juga terdapat lima unit komputer yang digunakan sebagai *attacker*.



Gambar 3.47 Arsitektur Jaringan Untuk Pengujian

3.2.3 Perancangan *Rules*

Seperti yang sudah dijelaskan sebelumnya pada bagian *rules* snort, *rules* pada snort memiliki dua bagian yaitu *rule header* dan *rule option*. Sebelum membuat rule snort maka perlu dibuat perancangan, struktur perancangan tersebut diambil dari hasil analisis serangan atau ancaman yang mungkin terjadi. Beberapa hal yang harus diketahui sebelum membuat *rules* yaitu :

1. Aksi yang akan dilakukan jika terjadi penyerangan (*Rule action*).
2. Protokol yang digunakan *attacker* dalam penyerangan (tcp, udp, icmp).
3. *Message* yang akan ditampilkan jika terjadi penyerangan.
4. Kemungkinan penyerangan dilakukan oleh siapa ? dari dalam atau luar jaringan (*source address*).
5. Kemungkinan penyerangan dilakukan ke siapa ? *IP address* target (*destination address*).
6. Kemungkinan *port* yang akan diserang (*port number*).
7. Paket data seperti apa yang dikirimkan *attacker* (*Payload*).

Pada penelitian ini akan dirancang beberapa *rule* snort untuk mendeteksi SQL *injection*, *Cross Site Scripting*, *Port Scanning*, DDOS.

3.2.3.1 Perancangan Rule SQL *injection*

Dari hasil analisis serangan SQL *injection* dapat disimpulkan, serangan ini terjadi akibat kelemahan dari layer aplikasi. Dengan merujuk kepada analisis serangan SQL *injection*, maka dapat dibuatkan *rule* untuk mendeteksi serangan ini, maka perancangan yang dilakukan adalah seperti pada Tabel 3.30.

Tabel 3.30 Perancangan Rule SQL *Injection*

No	Tipe	Isi
1	Aksi yang dilakukan	Alert dan Log
2	Protokol	TCP
3	Message Alert	Ada Penyerangan SQL injection !
4	Source Address	Internal dan eksternal (any)
5	Destination Address	Web server
6	Port Number	80 (HTTP), 443 (HTTPS)
7	Payload	%27 (simbol kutip 1 (‘)), UNION, SELECT, LIMIT, #, --, or 1=1

Contoh penulisan rule untuk SQL injection yaitu

```
alert tcp any any -> 192.168.165.1 80 (msg:"Serangan
SQL injection : Memasukan kata "union" pada URL";
content:"union";http_uri; nocase; sid:1000012;)
```

Rule tersebut digunakan untuk mendeteksi serangan apabila ada client yang menginputkan kata union pada url. *Rule* ini memiliki *signature id* 1000012.

3.2.3.2 Perancangan *Rule Cross Site Scripting*

Dari hasil analisis serangan *Cross Site Scripting* dapat disimpulkan, serangan ini terjadi akibat kelemahan dari layer aplikasi. Dengan merujuk kepada analisis serangan *cross site scripting* , maka dapat dibuatkan rule untuk

mendeteksi serangan ini, maka perancangan yang dilakukan adalah seperti pada Tabel 3.31.

Tabel 3.31 Perancangan *Rule Cross Site Scripting*

No	Tipe	Isi
1	Aksi yang dilakukan	Alert dan Log
2	Protokol	TCP
3	Message Alert	Cross-site scripting attempt
4	Source Address	Internal dan eksternal (any)
5	Destination Address	Web server
6	Port Number	80 (HTTP), 443 (HTTPS)
7	Payload	%2C => Simbol '<' %3E => Simbol '>' [a-z0-9\%] => angka dan huruf didalam tag html

Contoh penulisan rule untuk XSS yaitu :

```
alert tcp any any -> 192.168.165.1 80 (msg:"Serangan XSS : Memasukan tag html pada URL"; content:"<";http_uri; nocase; sid:1100012;)
```

Rule tersebut digunakan untuk mendeteksi serangan apabila ada *client* yang menginputkan tag html pada url. Rule ini memiliki signature id 1100012.

3.2.3.3 Perancangan Rule *Port Scanning*

Dari hasil analisis mekanisme *port scanning* dapat disimpulkan bahwa metode ini menggunakan beberapa paket-paket dan protokol yang berbeda, maka dalam perancangan rule deteksi terhadap mekanisme *port scanning* ditambahkan *syntax flag* dan *flow* untuk mendeteksi jenis flag yang dikirim. *Syntax flow* memungkinkan rule diterapkan pada arah tertentu dari alur trafik. Perancangan untuk mendeteksi mekanisme *port scanning* dapat dilihat pada Tabel 3.32.

Tabel 3.32 Perancangan Rule *Port Scanning*

No	Tipe	Isi
1	Aksi yang dilakukan	Log
2	Protokol	TCP, UDP
3	Message Log	Ada yang melakukan Port Scanning !
4	Source Address	Internal dan eksternal (any)
5	Destination Address	Internal
6	Port Number	Any Port
7	Payload	Flow : stateless Flags : FPU,12

3.2.3.4 Perancangan Rule DDOS

Dari hasil analisis penyerangan DDOS dapat disimpulkan bahwa metode ini menggunakan beberapa paket-paket dan protokol yang berbeda, maka dalam perancangan *rule* deteksi terhadap mekanisme port scanning ditambahkan *syntax flag* dan *flow* untuk mendeteksi jenis flag yang dikirim. Perancangan untuk mendeteksi penyerangan DDOS dapat dilihat pada Tabel 3.33.

Tabel 3.33 Perancangan Rule DDOS

No	Tipe	Isi
1	Aksi yang dilakukan	Alert dan Log
2	Protokol	TCP, UDP, ICMP
3	Message Alert	Ada Penyerangan DDOS !
4	Source Address	Internal dan eksternal (any)
5	Destination Address	Web server

No	Tipe	Isi
6	Port Number	any
7	Payload	Flow : stateless Size : packet size < 10000 byte Flags :S,12 Ping, pong, hello, AAAAAA, dan lain-lain tergantung jenis tools yang digunakan attacker.

3.2.4 Perancangan Proses Pengujian dan *Assesment*

Proses pengujian dan penilaian dilakukan dengan menyerang ke *server web* tersebut dengan beberapa metode penyerangan yaitu *SQL injection*, *Cross Site Scripting, DDOS dan Port Scanning*. Adapun parameter sebagai tolak ukur pengujian yaitu :

1. Kecepatan akses ke *web server* setelah *intrusion detection* diterapkan.
2. Ketepatan dalam pendeteksian serangan.

3.2.4.1 Perancangan Pengujian dan Penilaian Kecepatan Akses

Proses pengujian untuk mengukur kecepatan akses *server* dari *client* setelah *intrusion detection system* diterapkan yaitu dengan melakukan pencatatan waktu akses menggunakan FTP *client* dan *time* pada aplikasi ping dengan beberapa cara seperti pada Tabel 3.34.

Tabel 3.34 Form Pengujian dan Penilaian Kecepatan Akses

No	Cara	Waktu (milidetik)	
		Sebelum Terpasang IDS	Setelah Terpasang IDS
1	1 Client Ping paket normal	*(time aplikasi ping)*	*(time aplikasi ping)*

No	Cara	Waktu (milidetik)	
		Sebelum Terpasang IDS	Setelah Terpasang IDS
2	1 Client Ping paket 65000 *bytes*	*(time aplikasi ping)*	*(time aplikasi ping)*
4	2 Client Ping paket 65000 bytes secara bersamaan.	*(time aplikasi ping)*	*(time aplikasi ping)*
5	1 Client Upload File Melalui FTP	*(time app FTP)*	*(time app FTP)*
6	1 *Client* Melakukan Ping paket 65000 *bytes*, 4 *Client* Melakukan *Upload file*. Secara bersamaan.	*(time app FTP)dan time aplikasi ping)*	*(time app FTP)dan time aplikasi ping)*

Dari hasil pengukuran waktu pengujian dapat dihitung selisih waktu ketika setelah IDS terpasang dan sebelum IDS terpasang. Maka akan didapatkan kesimpulan apakah penerapan IDS ini mempengaruhi kecepatan akses terhadap *web server.*

3.2.4.2 Perancangan Pengujian Ketepatan Deteksi Serangan

Proses pengujian yang dilakukan untuk mengukur tingkat ketepatan atau akurasi dari *rule* yang dirancang yaitu dengan melakukan beberapa pengujian seperti pada Tabel 3.35 dan Tabel 3.36.

Tabel 3.35 Form Pengujian Deteksi Serangan SQL injection dan XSS

No	Jenis Serangan	Status
1	*SQL injection* : Mendapatkan nama database dari *web server*	*(Terdeteksi/Tidak Terdeteksi)*
2	*SQL injection* : Mendapatkan record-record dari *database*	*(Terdeteksi/Tidak Terdeteksi)*
3	*SQL injection* : Login ke webserver	*(Terdeteksi/Tidak Terdeteksi)*

No	Jenis Serangan	Status
4	*XSS :* Memasukkan tag html pada *web server*	*(Terdeteksi/Tidak Terdeteksi)*

Tabel 3.36 Form Pengujian Deteksi Serangan *Port Scanning* dan DDOS

No	Jenis Serangan	Status
1	*Port Scanning* : Menggunakan aplikasi nmap	*(Terdeteksi/Tidak Terdeteksi)*
2	*DDOS (Ping Large Data)*	*(Terdeteksi/Tidak Terdeteksi)*

3.2.5 Perancangan Aplikasi Monitoring IDS

Aplikasi ini berfungsi sebagai *user interface* antara IDS snort sebagai sarana untuk mempermudah administrator dalam memonitoring dan mengkonfigurasi snort. Aplikasi ini selanjutnya diberi nama snortbow.

3.2.5.1 Perancangan Basis Data

Perancangan basis data ini merupakan perancangan struktur tabel yang akan digunakan pada aplikasi snortbow untuk menyimpan data hasil *log* dari hasil deteksi aplikasi snort dan beberapa data penunjang lainnya yang dijadikan konfigurasi oleh aplikasi snortbow. Berikut ini perancangan struktur tabel dari snortbow.

1. Struktur table alert

   Table ini digunakan untuk menyimpan data berupa hasil parsing dari file alert yang dimiliki oleh aplikasi snort.

Tabel 3.37 Struktur Table Alert

Nama Field	Tipe Data	Ukuran	Kunci	Keterangan
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

2. Struktur table email

   Table ini digunakan untuk menyimpan id email yang nantinya akan dikirim email alert oleh aplikasi snortbow.

Tabel 3.38 Struktur Table Email

Nama Field	Tipe Data	Ukuran	Kunci	Keterangan
e_name	varchar	30	-	Not Null
e_mail	varchar	50	Primary Key	Not Null

3. Struktur table information

   Table ini digunakan untuk menyimpan informasi mengenai aplikasi snortbow, seperti waktu terakhir update data, waktu terakhir penyerangan, dan lain-lain.

Tabel 3.39 Struktur table information

Nama Field	Tipe Data	Ukuran	Kunci	Keterangan
snort_active	varchar	1	-	Not Null
last_update	datetime	-	-	Not Null
last_attack	datetime	-	-	Not Null
last_email	datetime	-	-	Not Null

4. Struktur table rule_detail

   Table ini digunakan untuk menyimpan data berupa id dan rule yang dimiliki oleh snort.

Tabel 3.40 Struktur Table rule_detail

Nama Field	Tipe Data	Ukuran	Kunci	Keterangan
id	int	11	-	Not Null
rule	text	-	-	Not Null
file_rule	int	11	-	Not Null

5. Struktur table rule_file

   Table ini digunakan untuk menyimpan data dari nama file rule yang digunakan snort.

Tabel 3.41 Struktur Tabel rule_file

Nama Field	Tipe Data	Ukuran	Kunci	Keterangan
id_file_rule	Int	11	Primary Key	Not Null
file_rule	varchar	50	-	Not Null

6. Struktur table sum_of_attack

   Table ini digunakan untuk menyimpan data dari jumlah penyerangan perhari.

Tabel 3.42 Struktur Table sum_of_attack

Nama Field	Tipe Data	Ukuran	Kunci	Keterangan
datetime	datetime	-	-	Not Null
attack	int	11	-	Not Null

7. Struktur table user

   Table ini digunakan untuk menyimpan username dan password untuk melakukan proses login.

Tabel 3.43 Struktur Table User

Nama Field	Tipe Data	Ukuran	Kunci	Keterangan
user	varchar	30	-	Not Null
password	varchar	50	-	Not Null
last_login	datetime	-	-	Not Null

3.2.5.2 Perancangan Antarmuka Pengguna

Antarmuka merupakan tampilan dari suatu program aplikasi yang berperan sebagai media komunikasi atau sarana dialog antara program dengan pengguna. Antarmuka yang baik adalah antarmuka yang mudah dipahami dan digunakan oleh pengguna. Perancangan antarmuka mendeskripsikan rencana tampilan dari setiap *form* yang akan digunakan pada aplikasi yang akan dibangun. Perancangan untuk aplikasi pada penelitian ini yaitu :

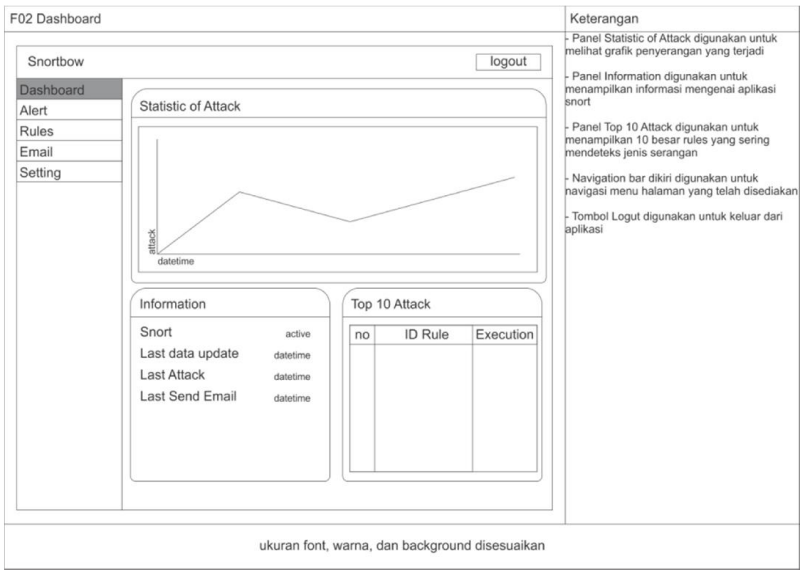
1. Perancangan antarmuka halaman login

   Halaman ini digunakan untuk melakukan otentifikasi user dan password sebelum masuk ke dalam aplikasi. Perancangan halaman login digambarkan pada Gambar 3.48.



Gambar 3.48 Perancangan Halaman Login

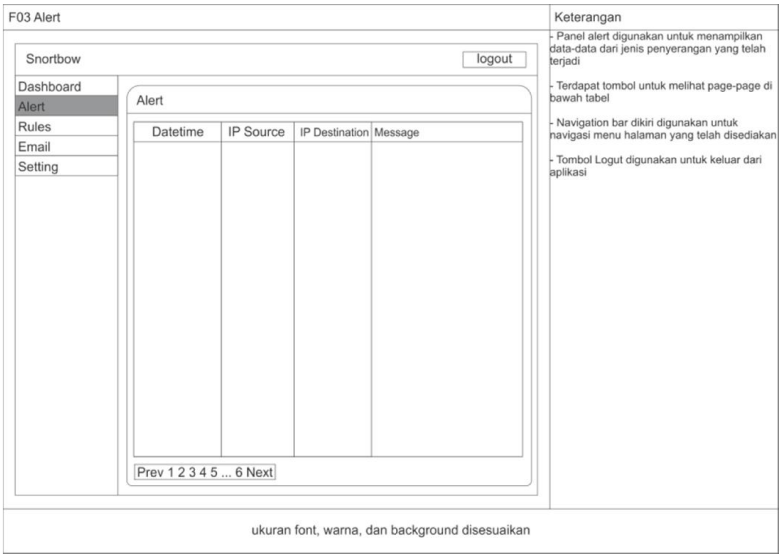
2. Perancangan antarmuka halaman dashboard
   Halaman ini digunakan untuk melihat grafik dan kesimpulan informasi-informasi.



Gambar 3.49 Perancangan Halaman Dashboard

3. Perancangan antarmuka halaman alert

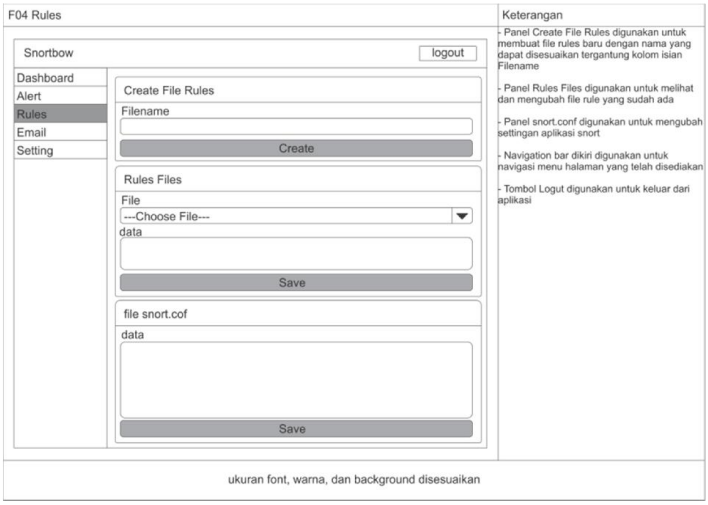
Halaman ini digunakan untuk melihat serangan-serangan yang telah di-log oleh snort.



Gambar 3.50 Perancangan Halaman Alert

4. Perancangan antarmuka halaman rules

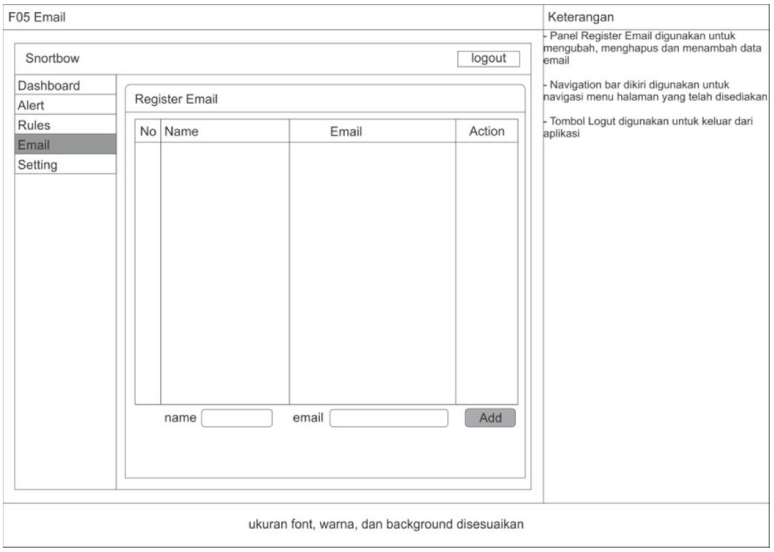
   Halaman ini digunakan untuk menambah dan mengubah file rules, pada halaman ini juga terdapat tab list untuk melihat daftar rule pada setiap file.



Gambar 3.51 Perancangan Halaman Rules

5. Perancangan antarmuka halaman email

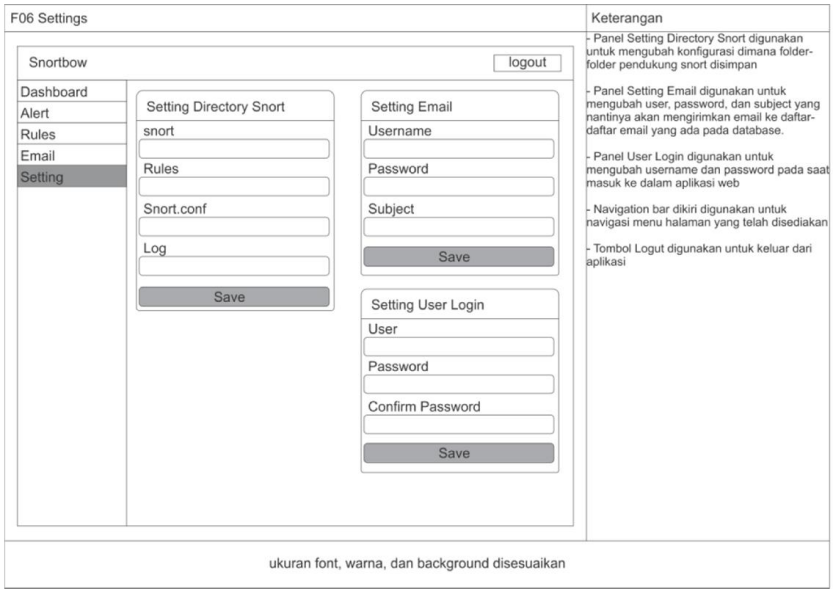
   Halaman ini digunakan untuk melihat, menghapus, mengubah dan menambah daftar email yang nantinya alamat email tersebut digunakan sebagai penerima alert yang dikirim oleh server IDS.



Gambar 3.52 Perancangan Halaman Email

6. Perancangan antarmuka 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.



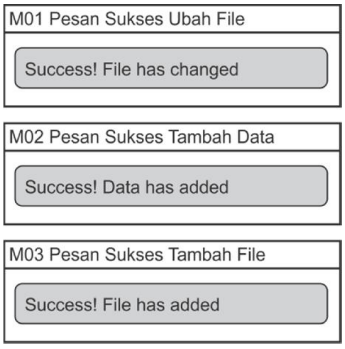
Gambar 3.53 Perancangan Halaman Settings

3.2.5.3 Perancangan Pesan

Pesan digunakan sebagai sarana untuk menginformasikan suatu tindakan, apakah tindakan tersebut berhasil atau gagal. Pesan juga dirancang sebagai media konfirmasi untuk meminimalisir kesalahan pada pengguna. Berikut ini merupakan gambar-gambar dari perancangan pesan pada aplikasi ini.

1. Perancangan Pesan Berhasil

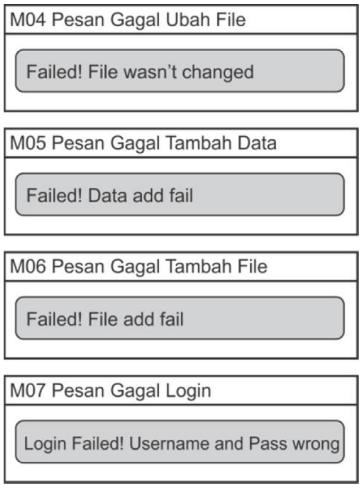
   Pesan ini digunakan sebagai notifikasi jika suatu perintah dari pengguna berhasil dieksekusi dengan benar.



Gambar 3.54 Perancangan Pesan Sukses

2. Perancangan Pesan Gagal

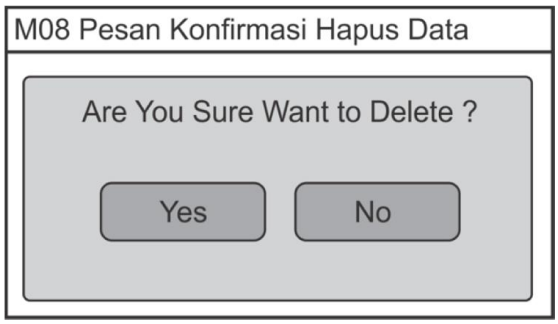
   Pesan ini digunakan sebagai notifikasi jika suatu perintah dari pengguna tidak berhasil dieksekusi dengan benar.



Gambar 3.55 Perancangan Pesan Gagal

3. Perancangan Pesan Konfirmasi

   Pesan ini digunakan untuk menanyakan kembali aksi yang eksekusi oleh pengguna.



Gambar 3.56 Perancangan Pesan Konfirmasi

4. Perancangan Pesan Pada Email Alert

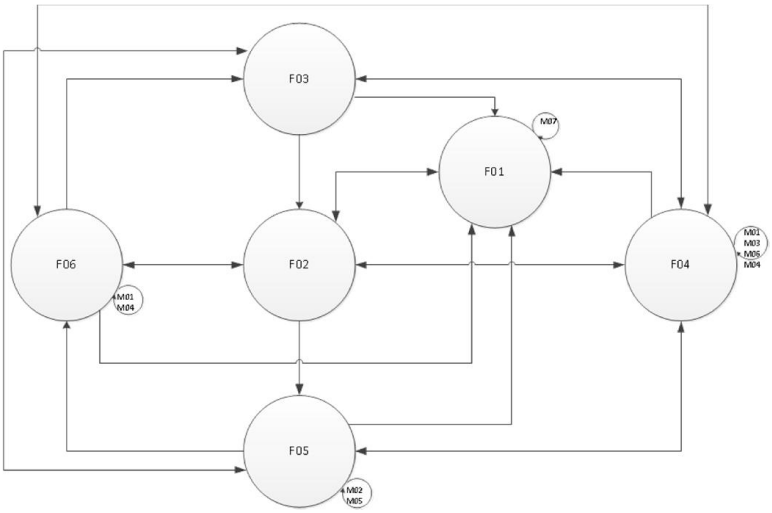
   Pesan ini digunakan sebagai notifikasi kepada email-email yang terdaftar di dalam database snortbow.

Tabel 3.44 Perancangan Pesan Email Alert

No	Parameter	Isi
1	From	Dikirim dari email dengan account gmail
2	To	Email-email yang telah didaftarkan
3	Subject	Disesuaikan dengan konfigurasi pada halaman setting, dibelakang subjek ditambahkan datetime
4	Data	Data Serangan dari file Alert berupa IP address source, ip address destination dan message dari serangan tersebut.

3.2.5.4 Perancangan Jaringan Semantik

Perancangan jaringan semantik untuk aplikasi ini dapat dilihat pada Gambar 3.57.



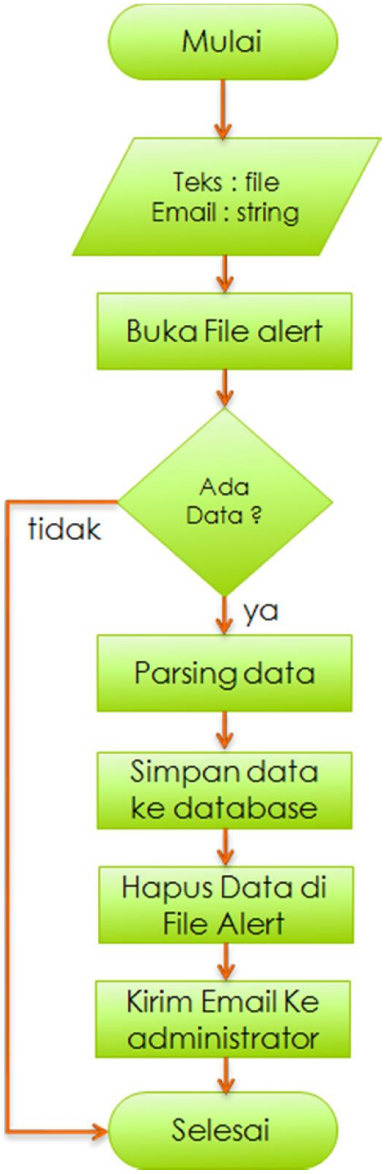
Gambar 3.57 Perancangan Jaringan Semantik

3.2.5.5 Perancangan Prosedural

Perancangan prosedural merupakan Perancangan yang dilakukan untuk menetapkan detail algoritma yang akan dinyatakan ke dalam suatu sistem. Adapun Perancangan prosedural untuk aplikasi berbasis web yang akan dibangun yaitu :

1. Perancangan Prosedur Parsing Alert

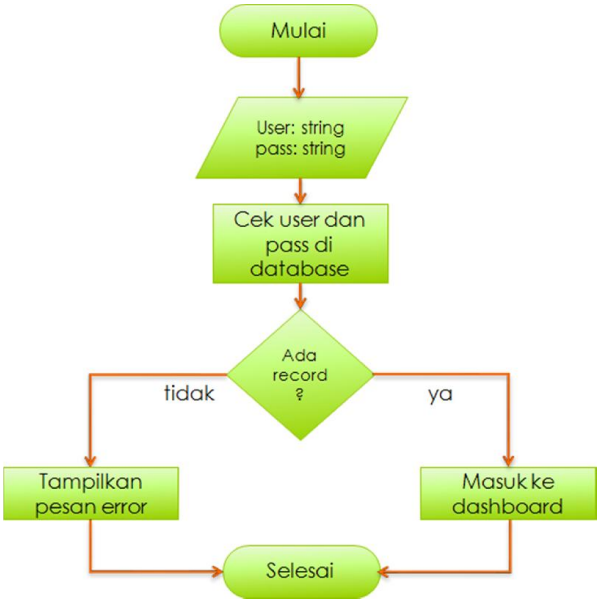
   Prosedur ini digunakan untuk melakukan parsing dari file alert yang ada pada *log* aplikasi snort.



Gambar 3.58 Flowchart Proses Parsing Alert

2. Perancangan Proses Login

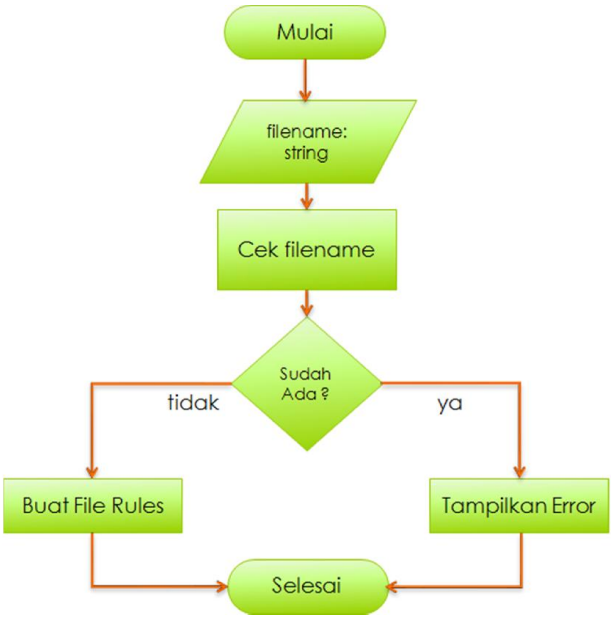
   Proses ini digunakan pada saat pengguna akan masuk ke dalam aplikasi.



Gambar 3.59 Flowchart Proses Login

3. Perancangan Proses Tambah File Rules

   Proses ini digunakan untuk menambahkan file rule pada snort.



Gambar 3.60 Flowchart Proses Tambah File Rules

4. Perancangan Proses Ubah Data File

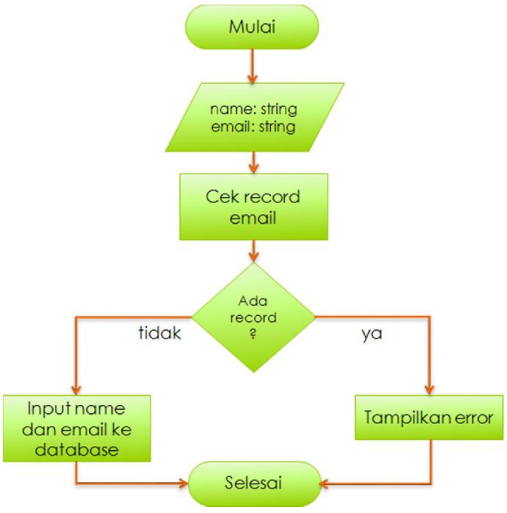
   Proses ini digunakan untuk mengubah konfigurasi pada aplikasi snorbow.



Gambar 3.61 Flowchart Proses Ubah Data File

5. Perancangan Proses Tambah Data Email

   Proses ini digunakan untuk menambah daftar email pada database.



Gambar 3.62 Flowchart Proses Tambah Data Email