e-ISSN: 2252-7036
Vol.9 No.1 2024: 34-40

Network – Based Intrusion Detection System Menggunakan Snort Terhadap Smurf Attack pada Jaringan Komputer

Masduki Khamdan Muchamad[1], M. Alza Adriansyah[2], Ramzi Adriman[3]

Jurusan Teknik Elektro dan Komputer, Universitas Syiah Kuala
Jln. Teuku Nyak Arief Darussalam, Banda Aceh, Aceh, 23111, Indonesia Aceh
masduki@usk.ac.id
alza.adriansyah2000@gmail.com
ramzi.adriman@usk.ac.id

Abstrak— **Pemantauan jaringan dengan melakukan pembacaan terhadap setiap paket jaringan yang masuk ataupun keluar dari setiap *device,* alat pendeteksi keadaan *traffic* jaringan menjadi hal terpenting dalam mengelola sebuah jaringan, Snort NIDS dalam hal ini dapat berperan sebagai mesin utama untuk melakukan pemantauan terkait jaringan yang sedang digunakan. Dalam hal ini pendeteksian paket jaringan difokuskan terhadap paket serangan *smurf attack*. Snort kemudian dikombinasikan dengan BASE untuk menghasilkan *alert* dari setiap paket serangan yang diluncurkan, didalam proses pendeteksian serangan yang masuk ke dalam sebuah jaringan, snort menghasilkan *log* yang berisi setiap paket serangan yang berhasil di deteksi, yang kemudian *log* tersebut diterjemahkan oleh Barnyard2 dan disimpan kedalam sebuah *database* MySQL, hasil tesebut kemudian dibaca oleh BASE untuk menampilkan setiap *alert* yang berhasil di deteksi, hal ini ditujukan agar snort dapat merekam setiap *alert* yang berhasil di deteksi agar tidak terjadinya kesalahan pendeteksian, mendapatkan *alert* yang tepat, juga untuk mengetahui dampak dari serangan yang dilakukan, dalam pengujian didapatkan hasil pemantauan penggunaan CPU pada komputer korban hingga mencapai 96.3% pada titik tertinggi dan jumlah paket serangan yang diterima sebesar 20 paket pada serangan pertama dan 120 paket pada serangan kedua, terjadi *lag* pada komputer korban yang disebabkan oleh penggunaan *resource* cpu yang besar diakibatkan dari serangan tersebut.**

Kata Kunci*— *NIDS, Snort, IDS, BASE, Keamanan jaringan, Smurf attack

I. PENDAHULUAN

Snort berperan sebagai *Intrusion Detection System* (IDS), snort memiliki fungsi mendeteksi serangan *smurf* yang di luncurkan, dengan perancangan *snort rules* untuk mendeteksi serangan *smurf* yang diluncurkan oleh komputer penyerang, data penyerangan tersebut di simpan didalam log hasil rekaman penyerangan yang telah di deteksi oleh Snort yang di terapkan sebagai *Intrusion Detection System* (IDS). *Intrusion Detection System* (IDS) dapat secara langsung menampilkan *"alert"* ataupun peringatan kepada pengguna jika terjadi aktivitas ataupun perilaku yang mencurigakan pada sebuah jaringan komputer. IDS berkemampuan untuk melakukan penentuan jenis serangan yang dilakukan kepada jaringan komputer, dengan menerapkan *snort rules* maka setiap serangan yang masuk, dapat dideteksi oleh sistem Snort tersebut sehingga dapat menentukan jenis serangan yang merugikan yang dilakukan ke sebuah jaringan komputer yang sedang dipakai [1].

Snort adalah *tools* yang memiliki kegunaan untuk melakukan pencegahan terhadap intrusi ataupun serangan pada suatu jaringan, dalam penggunaannya snort dapat membuat *logging* yang berisi paket-paket jaringan juga analisis dari *traffic* pada suatu jaringan secara *real time*, snort adalah kombinasi dari analisis protokol pada sistem dan pendeteksi penyusupan pada sistem atau yang sering disebut dengan IDS, yang baik dalam mendeteksi setiap serangan yang dilakukan kepada sebuah jaringan [2].

Cara kerja snort memiliki fokus pada *security packet sniffing*, snort juga memiliki fitur yakni *payload inspection* yaitu snort dapat melakukan analisis pada *payload rule set* yang telah ditetapkan. Ketika paket dikirimkan melalui jaringan tersebut, snort langsung mencocokkan dengan *rules* yang telah ditetapkan, jika terindikasi bahwa terdapat serangan dalam paket tersebut, maka paket tersebut disimpan di dalam log snort [3]

Dalam pengoperasiannya, snort membentuk *logging* dari paket-paket yang dianalisis secara *real time*, dalam artian snort membuat semacam pencatatan *alert* yang dicatat ke dalam sebuah database yang telah ditetapkan oleh administrator jaringan [4].

Proses penyerangan *smurf attack* dilakukan dengan diawali penyerang yang mengirimkan pesan request kepada perantara, yang kemudian pesan ini akan di *broadcast* kepada semua perangkat perantara yang ada, ketika pesan broadcast yang berupa ICMP *request* tersebut telah diterima oleh perantara, maka semua yang menerima pesan *broadcast* tersebut akan memberi balasan atas *request* tersebut, yang mana tentu saja akan membanjiri target yang ingin di serang [5].

Snort *rules* merupakan sebuah komponen yang berfungsi untuk melakukan pengklasifikasian terhadap segala potensi intrusi yang ada ataupun sebuah ancaman yang dapat

membahayakan jaringan, sangat diperlukan ketelitian dalam membuat snort *rules* untuk menghindari terjadinya *false alarm* yaitu kondisi dimana timbulnya *alert* pada saat tidak terjadi sebuah serangan ke jaringan [6].

Aktivitas yang dinilai berbahaya bagi jaringan yang sedang digunakan dan dapat mengancam sistem akan memicu untuk dikirimkannya pesan peringatan kepada *administrator* jaringan, *administrator* membuat snort *rules* tersebut didalam

sebuah teks yang mana didalamnya terdapat deskripsi mengenai peringatan dan juga konfigurasi dari snort itu sendiri, sehingga snort dapat mendeteksi dan melakukan pemantauan terhadap lalu lintas aktivitas jaringan [7]

BASE dapat diimplementasikan menjadi sebuah *web* yang dapat melakukan berbagai macam analisa mengenai *intrusi* yang terjadi dalam sebuah jaringan, log yang telah didapatkan dari IDS dapat dianalisa di dalam BASE, dengan berbasis PHP dengan *interface* yang mudah dimengerti, dapat mempermudah proses pengidentifikasian sebuah *intrusi* yang terjadi di dalam sebuah jaringan, *intrusi* yang didapatkan oleh BASE berasal dari snort IDS yang telah mengumpulkan log - log aktivitas dalam sebuah jaringan, NIDS yang diintegrasikan dengan BASE yang digunakan untuk menampilkan serangan yang telah dideteksi oleh snort *rules* dengan tampilan *web interface* dari BASE sangat membantu *administrator* dalam memantau *traffic* jaringan [8].

BASE merupakan sebuah antarmuka yang dapat membantu melakukan analisa terhadap intrusi - intrusi yang telah di deteksi pada sebuah jaringan, BASE juga merupakan sebuah program yang berbasis PHP yang dapat melakukan pemrosesan terhadap *database* hasil dari *security event* yang didapatkan dari IDS. Yang menjadi fungsi utama dari BASE sendiri adalah sebagai *Graphic User Interface* (GUI), yang dapat mempermudah bagi *administrator* jaringan dalam memantau log aktivitas jaringan yang dicatat didalam sebuah *database* [9].

II. METODE PENELITIAN

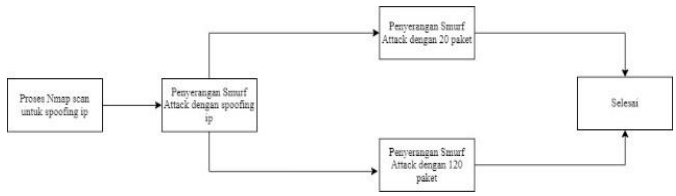
Tahap-tahap penelitian antara lain yakni Studi literature, Instalasi snort, Konfigurasi snort rules, Konfigurasi Barnyard2, Konfigurasi MySQL, Konfigurasi BASE, pengujian system, kemudian penulisan laporan.

Bahan yang dikumpulkan pada awal penelitian yakni referensi terkait dengan sistem NIDS dengan snort, konfigurasi yang dilakukan, juga terkait dengan beberapa plugin yang dapat dihubungkan dengan snort NIDS.

A. Rancangan Konseptual

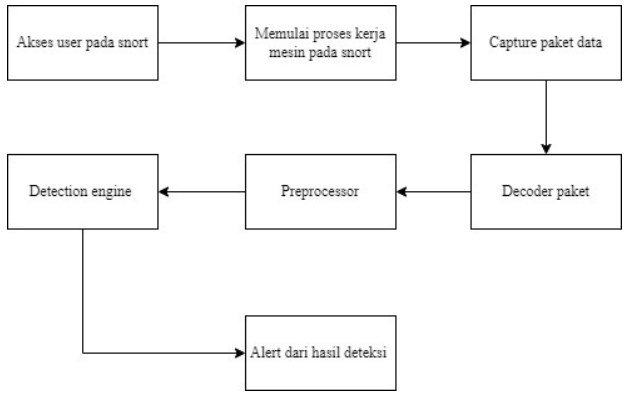
Merupakan tahapan awal dalam proses perancangan pengujian sistem keamanan jaringan ini, pada tahapan ini setiap konsep dari ide di definisikan dan dijelaskan dengan detail. Tahapan ini untuk memperjelas terkait dengan pengujian fungsi dan fitur dari sistem yang dirancang.

1) Penyeranngan Smurf Attack: Proses penyerangan Smurf Attack dilakukan dengan diawali Nmap scan pada jaringan, hal ini ditujukan untuk melihat jumlah host yang up di dalam sebuah jaringan juga untuk menentukan komputer korban yang akan dilakukan spoofing ip terhadapnya, penyerangan dilakukan dengan pengiriman paket smurf attack sebesar 20 paket dan 120 paket pada dua buah jaringan yang berbeda.



Gambar. 1 Penyerangan Smurf Attack

2) Mendapatkan Alert: User harus melakukan akses ke snort untuk menggunakan snort tersebut, kemudian akan dilakukan penjalanan proses awal yakni memulai kerja sistem pendeteksi, kemudian paket data dari jaringan yang sedang diawasi akan di *capture* ataupun diambil, dengan telah selesainya *capture* data maka paket akan di *decode* dengan tujuan mempersiapkan paket untuk diproses menuju ke *detection engine*, setelah itu paket akan masuk ke tahap *preprocessor* yakni penyusunan paket data, dalam hal ini snort dapat mendefragmentasi paket yang ada, yang berguna untuk mempermudah *detection engine* dalam memeriksa apakah terdapat paket yang berasal dari penyusup atau tidak, *detection engine* memiliki peran yang sangat penting yakni mendeteksi apakah terdapat serangan dari penyusup ataupun tidak, *detection engine* menggunakan *rules* yang ada untuk mencocokkan dengan paket data yang diterima, dan dari semua paket data tersebut, apabila terjadi *hit* ataupun *match* antara paket data dengan *rules* yang ada, maka paket data tersebut akan dikategorikan menjadi sebuah ancaman ataupun serangan yang dilakukan pada sebuah jaringan, setelah paket data tersebut berhasil di deteksi dan terindikasi bahwa merupakan sebuah jaringan, maka akan muncul *alert* ataupun pemberitahuan terjadinya sebuah jaringan yang di *trigger* oleh hasil deteksi tersebut.

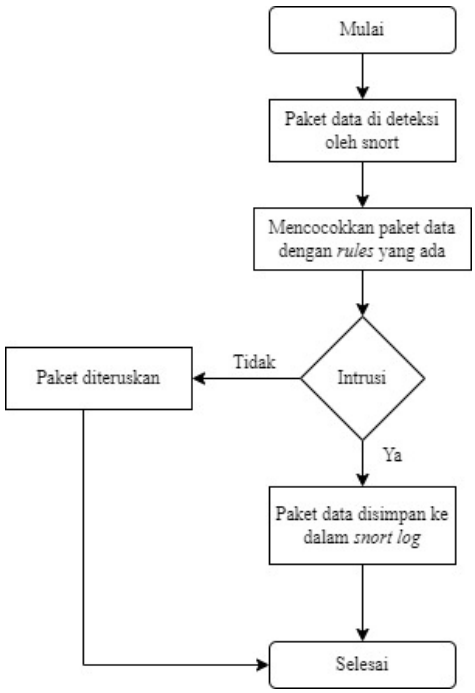


Gambar. 2 Proses Mendapatkan Alert

3) Pendeteksi Smurf Attack Dengan Snort Rules: Diagram tersebut menjelaskan bahwa paket data mulanya akan masuk ke dalam lalu lintas jaringan dan akan direkam oleh snort, kemudian setiap paket data yang masuk dilakukan pencocokan paket data dengan snort *rules* yang ada, jika paket

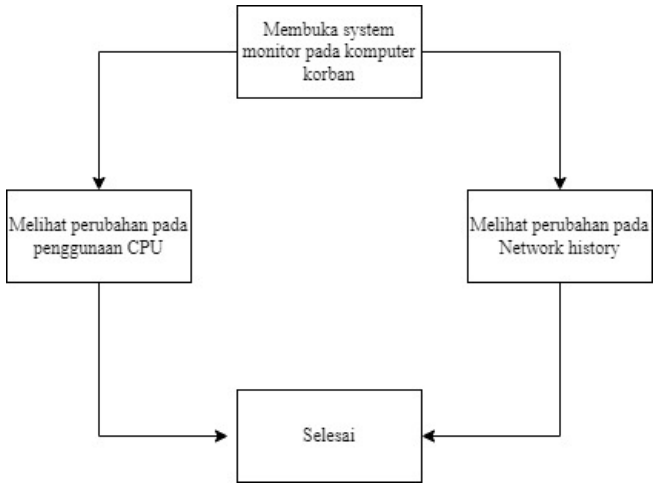
e-ISSN: 2252-7036
Vol.9 No.1 2024: 34-40

data tersebut merupakan sebuah intrusi maka snort menyimpan paket tersebut ke dalam *snort log* jika sebaliknya maka paket tersebut tidak disimpan ke dalam *snort log.*



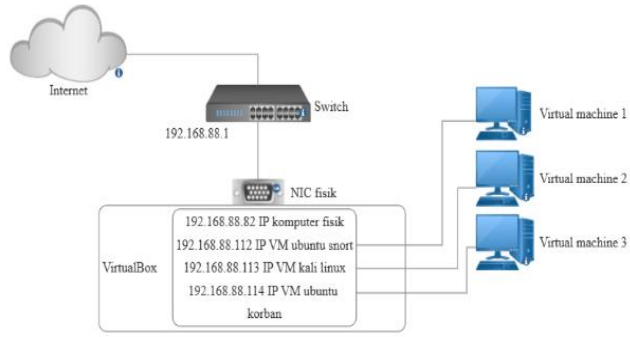
Gambar. 3 Pendeteksi Serangan Dengan Snort Rules

4) Mengetahui Efek Serangan Yang Dilakukan: Untuk mengetahui dampak serangan yang dilakukan terhadap komputer korban, dilihat dengan melakukan pemantauan pada system monitor milik VM korban lalu melihat perubahan terkait dengan penggunaan CPU dan Network pada saat sebelum serangan dengan sesudah serangan dilakukan.



Gambar. 4 Pengecekan Dampak Serangan Smurf Attack

5) Topologi Yang Digunakan: Pengkonfgurasian jaringan dilakukan pada *software virtualbox*, jenis jaringan yang digunakan adalah *bridget network*, dimana mode ini menghubungkan antara adapter jaringan VM ke jaringan fisik yang sedang digunakan, dengan memilih jenis *bridget network* untuk digunakan, maka setiap VM yang dikonfigurasikan dengan jenis tersebut menjadi bagian daripada jaringan yang sedang digunakan oleh komputer fisik, jika *host* mendapatkan IP address DHCP maka VM juga dapat beroperasi seperti itu. Dalam hal ini setiap VM dapat terkoneksi dengan VM yang lainnya juga kepada semua komputer fisik yang berada dalam jaringan internet fisik. Bridget adapter memberikan *ip address* kepada semua VM yang ada jika VM tersebut terkonfigurasi dengan *bridget adapter.*



Gambar. 5 Topologi yang digunakan

III. HASIL DAN PEMBAHASAN

Berikut merupakan pembahasan mengenai hasil yang didapatkan dari penelitian yang telah dilakukan, yang berisi hasil rancangan dan pengujian dari sistem yang telah dibuat.

A. Hasil Rancangan

Berikut merupakan hasil rancangan yang didapat sesuai dengan penelitian yang telah dilakukan, hasil rancangan berupa sistem yang telah dibuat untuk menjalankan penelitian, berikut beberapa hasil rancangan sistem yang telah dibuat.

1) Hasil Instalasi Snort: Instalasi snort dilakukan didalam VM ubuntu *virtualbox*, dalam hal ini penginstalan snort didahului dengan mengunduh snort versi 2.9.20 dan juga snort daq versi 2.0.7 dari *website* resmi snort yang diunduh melalui terminal ubuntu file dan disimpan didalam direktori snort_src, berikut perintah yang dijalankan untuk menampilkan versi snort yang berhasil diinstall.

- #root@alza-VirtualBox: snort –V
  -*> Snort! <*.
  Version 2.9.20 GRE (Build 82)

Untuk menjalankan snort dalam mode NIDS maka diperlukan pengeditan file konfigurasi utama milik snort yang ada pada direktori /etc/snort dengan nama file snort.conf, dengan menjalankan snort dengan file ini sebagai argumen, menandakan bahwa snort dijalankan dalam mode NIDS, maka dilakukan pengeditan file snort.conf yang berada pada direktori /etc/snort yang dapat dilihat pada perintah dibawah ini,

pengkonfigurasian diawali dengan mengisi ip yang digunakan pada file konfigurasi tersebut.

- File: /etc/snort/snort.conf
  # Step #1: Set the network variables. For more information, see README.variabl$
  # Setup the network addresses you are protecting
  ipvar HOME_NET 192.168.88.0/24

2) Hasil Konfigurasi Snort Rules: Dilanjutkan dengan dilakukannya pengaktifan terkait dengan file local.rules yang berada pada direktori /etc/snort/rules/local.rules yang dimana pada file ini dilakukan penambahan *rules* yang menentukan *alert* pada snort, yang dilakukan dengan menghapus hash pada file konfigurasi utama snort yaitu snort.conf untuk mengaktifkan rules *local* yang digunakan.

- File: /etc/snort/snort.conf
  # Step #7: Customize your rule set
  # For more information, see Snort Manual, Writing Snort Rules
  # NOTE: All categories are enabled in this conf file
  #site specific rules
  include $RULE_PATH/Local.rules

Kemudian dilakukan pembuatan Snort *rules* yang dibutuhkan untuk mencocokkan setiap paket yang terdeteksi oleh snort dan mengkategorikan paket – paket tersebut apakah menjadi sebuah ancaman untuk jaringan yang digunakan ataukah tidak, dalam hal ini snort rules dibuat untuk mengkategorikan setiap paket yang tergolong sebagai *smurf attack*, berikut *snort rules* yang telah dibuat.

- Alert tcmp any any > SHOME_NET any (msg: "SMURF attack detected"; {type: 8; Sid: 5000002; rev: 1; classtype:attempted-recon;)

3) Hasil Konfigurasi Barnyard2: Banyard2 dibutuhkan dalam hal menyimpan output yakni *alert* yang dihasilkan oleh snort ke dalam *database* MySQL, snort dikonfigurasikan untuk menghasilkan *output* dalam bentuk biner ke dalam sebuah folder yang kemudian folder tersebut dapat dibaca oleh barnyard2 terkait *event* yang datang tersebut dan menyimpannya ke dalam *database* MySQL. Penginstalan dilakukan dengan mengunduh file Barnyard2 melalui terminal ubuntu dan menginstall file *installer* tersebut.

- root@alza-VirtualBox:~# barnyard2 -V
  -*> Barnyard2 <*.
  Version 2.1.14 (Build 337)

4) Hasil Konfigurasi MySQL: Karena barnyard2 menyimpan setiap *alert* yang dihasilkan ke dalam *database* MySQL, maka untuk itu dibuat *database* yang dapat diakses oleh user 'snort', dan juga dilakukan pembuatan MySQL snort user password, berikut *database* yang dibuat yang dapat dilihat pada text tersebut.

- mysql> SHOW DATABASES;
  | Database
  | information_schema
  | mysql
  | performance_schema
  | snort
  | sys
  5 rows in set (0.03 sec)

Kemudian dilakukan pengecekan terhadap *database* MySQL apakah terdapat *alerts* yang di tulis oleh barnyard2 kedalam *database* MySQL, untuk mengecek database tersebut dijalankan perintah berikut

- root@alza-VirtualBox:/var/log/snort# mysql -u root -p -D snort -e "select count(*) from event"
  Enter password:
  | count(*) |
  | 101014 |

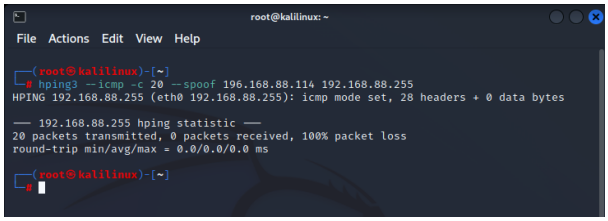
5) Hasil Konfigurasi BASE: Pada file konfigurasi tersebut yakni base_conf.php dilakukan pengeditan terkait BASE_urlpath yang merupakan lokasi url dari BASE itu sendiri, lalu DBLIB_path yang merupakan lokasi *library* DB kemudian juga pengeditan terkait nama dari *database* yang digunakan yakni alert_dbname, dan juga dilakukan pengeditan pada nama *host, user,* dan *password* yang digunakan pada *database* yang mana pada file konfigurasi tersebut disetting menjadi seperti perintah berikut

- File: base_conf.php
  $BASE_urlpath = '/base';
  $DBlib_path = '/var/adodb/';
  $alert_dbname= 'snort';
  $alert_host= 'localhost';
  $alert_user='snort';
  $alert_port = '';
  $alert_password = 'alza';

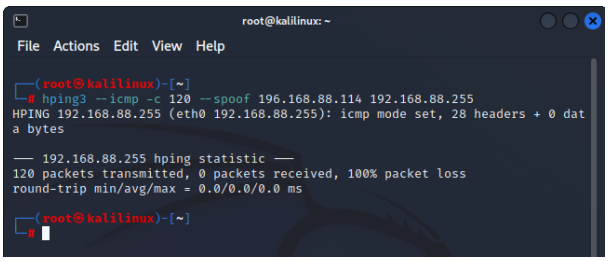
B. Penyerangan Smurf Attack

Peneliti melakukan penyerangan *smurf attack* dari VM kali linux yang ditujukan kepada jaringan yang sedang dipakai dengan ip yaitu 192.168.88.0/24 dengan menggunakan terminal kali linux untuk mengirimkan paket serangan *smurf attack* yang berupa ICMP *echo request* sebanyak 20 dan 120 paket tersebut, yang diawali dengan Nmap *scan* terhadap jaringan yang dipakai untuk melihat *host* yang ada dalam jaringan dan didapati bahwa terdapat 21 *host* yang up di jaringan tersebut

e-ISSN: 2252-7036
Vol.9 No.1 2024: 34-40



Gambar. 6 Smurf Attack Sebesar 20 Paket

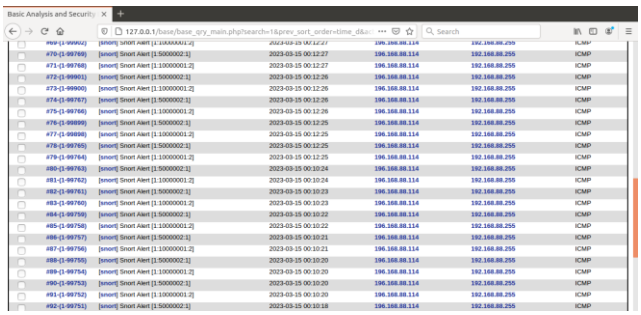


Gambar. 7 Smurf Attack Sebesar 120 Paket

C. *Snort Alert*

Setelah *smurf attack* dijalankan, snort mulai mengeluarkan *output* pada layar terkait dengan setiap paket – paket serangan yang berhasil di deteksi, snort langsung menampilkan setiap paket serangan tersebut ke layar, berikut gambaran snort pada ubuntu setelah terjadinya penyerangan *smurf attack*, dalam hal ini alert yang didapat ada 2 yakni alert pada terminal snort dan alert pada BASE

- 03/14-23:45:57.012586 [**] [1:5000002:1] SMURF attack detected [**] [Classification: Attempted Information Leak] [Priority:2] (ICMP) 196.168.88.114 -> 192.168 .88.255



Gambar. 8 BASE Alert

D. *Snort Log*

Setelah snort berhasil merekam setiap paket yang sesuai dengan *rules* yang ada, maka snort menyimpan setiap *alert* tersebut ke dalam snort *log* yang berada di direktori var/log/snort. Setiap snort *log* yang dihasilkan dapat menampilkan snort *alert*, karena isi dari setiap snort *log* tersebut merupakan snort *alert* yang berhasil disimpan, snort menyimpan setiap *alert* tersebut ke dalam snort *log* yang berada di direktori var/log/snort, berikut merupakan gambaran isi dari setiap log snort yang berhasil didapatkan

- WARNING: No preprocessors configured for policy 0.03/15-00:10:23.925501 196.168.88.114 192.168.88.255 ICMP TTL:64 TOS:x ID:26830 IpLen:20 Ognen:28 Type:8 Code:0 0:58133 Seq:4608 ECHO
  WARNING: No preprocessors configured for policy. 03/15-00:10:23.925501 196.168.88.114 192.168.88.255 ICMP TTL:64 TOS: 10:26830 IpLen:20 Ognen:28 Code:0 10:58133 Seq:4608 ECHO

E. *Runtime Packet Snort*

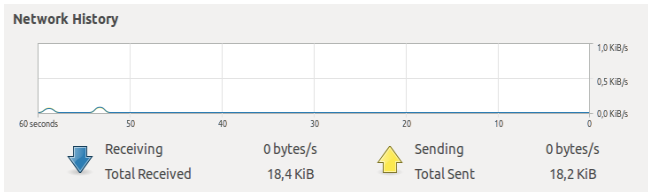
Terkait dengan detail paket serangan yang terekam pada log snort, terdapat informasi yang dapat dilihat yakni *run time* untuk packet *processing* yang dalam hal ini didapatkan selama 0.1936 detik dan dalam kurun waktu tersebut snort dapat mencatat sebanyak 40 paket yang terindikasi sebagai sebuah paket yang sesuai dengan *rules* yang telah dibuat

- Run time for packet processing was 0.1936 seconds
- Snort processed 40 packets

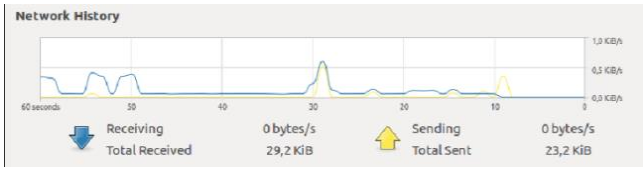
F. *Dampak Serangan Yang Ditimbulkan*

Pemantauan dilakukan pada *system monitor* komputer korban, dalam hal ini data yang didapatkan berasal dari *network history* dan CPU *history* pada komputer korban tersebut, hal ini ditujukan untuk mengetahui dampak yang ditimbulkan smurf attack pada komputer korban.

1) Network Usage: Terdapat perubahan yang menunjukkan adanya peningkatan pada penggunaan *network* yang dilihat dari nilai total received yang sebelumnya sebesar 18.4 KiB menjadi 29.2 KiB juga pada total sent yang sebelumnya sebesar 18.2 KiB menjadi 23.2 KiB, akan tetapi peningkatan yang terjadi tidak terlalu signifikan, kenaikan yang terjadi cukup sedikit.



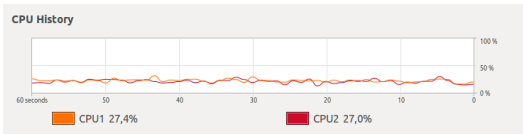
Gambar. 9 Network Usage Sebelum Serangan



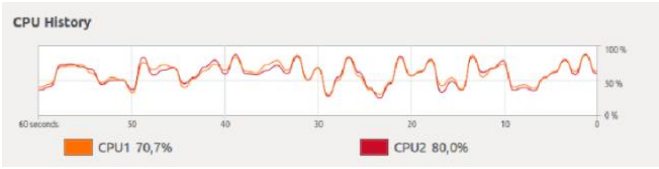
Gambar. 10 Network Usage Setelah Serangan

2) CPU USAGE: Dampak yang dihasilkan oleh *smurf attack* pada penggunaan CPU cukup banyak yang menyebabkan komputer terasa menjadi lebih berat dari sebelumnya yang dapat dilihat pada gambar yang tertera

diatas, *smurf attack* menghasilkan peningkatan yang signifikan yang menghasilkan dampak yang cukup besar untuk penggunaan CPU.



Gambar. 11 CPU Usage Sebelum Serangan



Gambar. 12 CPU Usage Setelah Serangan

G. Hasil Monitoring Sistem Snort NIDS

Pada Tabel I dapat dilihat bahwa dilakukan 4 buah serangan pada 2 buah jaringan yang berbeda pada jaringan pertama dengan ip 192.168.88.0/24 menggunakan jaringan LAN dan pada jaringan kedua dengan ip 192.168.175.0/24 menggunakan jaringan *wi-fi* dengan paket serangan yang dikirim dari VM kali linux yang merupakan sebuah penyerang dengan jumlah paket sebesar 20 dan 120 paket.

TABLE I
JUMLAH PAKET SERANGAN YANG DIKIRIM

	IP penyerang	***IP korban***	***IP jaringan***	***Paket yang dikirim***
1	192.168.88.113	192.168.88.114	192.168.88.0/24	20
2	192.168.88.113	192.168.88.114	192.168.88.0/24	120
3	192.168.175.17	192.168.175.144	192.168.175.0/24	20
4	192.168.175.17	192.168.175.144	192.168.175.0/24	120

H. Runtime Packet Snort

Pada Tabel II dapat dilihat bahwa paket yang diterima oleh sistem NIDS berjumlah 40, 240, 40, dan 240 paket, hal ini dikarenakan terdapat dua *rules* yang diaktifkan pada snort *rules* untuk mendeteksi setiap paket yang dikirim dalam jaringan, yakni ICMP *rules* dan ICMP *echo rules*, dikarenakan *smurf attack* merupakan sebuah serangan yang dilakukan dengan protocol ICMP maka sistem snort mengkategorikan paket tersebut menjadi paket dengan tipe yang sama yaitu ICMP akan tetapi snort mendeskripsikan perbedaan pembacaan antara kedua *rules* pada paket tersebut yang dapat dilihat pada hasil pembacaan dalam snort *log*, *rules* untuk mengkategorikan *smurf attack* mengkategorikan paket tersebut sebagai ICMP *echo request*, sedangkan *rules* untuk ICMP packet hanya mengkategorikan paket tersebut sebagai paket ICMP biasa, sehingga paket yang dikirimkan sebesar 20 dan 120 paket, diterima menjadi 40 dan 240 pada snort, berdasarkan data tersebut paket yang diterima dan paket yang berhasil dianalisis oleh snort memiliki persentase 100 % yang berarti snort secara akurat mendeteksi setiap paket yang masuk pada suatu jaringan yang sedang dimonitor yakni berupa paket ICMP, karena tidak ada paket yang tidak berhasil dianalisis. Kemudian untuk *run time packet* snort yang merupakan kecepatan snort untuk menampilkan snort *log* yang berhasil disimpan, didapat pada percobaan pertama selama 0.1936 *seconds* dengan paket berjumlah 40 paket, kemudian pada percobaan kedua selama 0.43321 *seconds* dengan paket berjumlah 120 paket, lalu pada percobaan ketiga dengan jumlah paket 40 paket memiliki *run time packet* snort selama 0.579 *seconds*, dan terakhir pada percobaan keempat dengan paket berjumlah 240 paket memiliki *run time packet* snort selama 0.43321 *seconds.*

TABLE III
RUN TIME PACKET SNORT

	Jumlah paket yang diterima oleh NIDS	***Jumlah paket yang berhasil dianalisis***	***Persentase paket tergolong sebagai ICMP***	***Run time packet snort***
1	40	40	100 %	0.1936 seconds
2	240	240	100 %	0.36034 seconds
3	40	40	100 %	0.579 seconds
4	240	240	100 %	0.43321 seconds

I. Hasil System Monitoring Pada Client

Berikut merupakan data pada *client* yang diambil pada saat sebelum terjadinya serangan dan sesudah terjadinya serangan, dalam hal ini dapat dilihat terdapat CPU *history* dan juga *network history* saat sebelum dan sesudah serangan dalam hal ini data disajikan dalam bentuk tabel seperti pada tabel III.

TABLE IIIII
PENGGUNAAN CPU PADA VM KORBAN

	Jumlah Paket serangan	**CPU sebelum serangan**		**CPU sesudah serangan**	
		CPU 1	***CPU 2***	***CPU 1***	***CPU 2***
1	20 paket	27.4 %	27.0 %	70.7 %	80.0 %
2	120 paket	21.2 %	24.3 %	54.1 %	96.3 %
3	20 paket	18.0 %	20.7 %	41.0 %	40.6 %
4	120 paket	15.4 %	13.0 %	49.5 %	69.2 %

Berdasarkan tabel diatas dapat dilihat bahwa penggunaan CPU pada VM korban terhadap 4 kali serangan yakni pada CPU 1 dan 2 mengalami peningkatan yang cukup signifikan pada CPU 1 dan CPU 2, disini terdapat 2 buah CPU *core* dikarenakan penambahan CPU *core* pada *virtualbox* untuk VM agar mengatasi *lag* ataupun *crash* pada saat menjalankan VM. Terjadi perbedaan pada saat pengiriman serangan *smurf attack* dengan 20 paket dan 120 paket, penggunaan CPU yang dihasilkan memiliki nilai lebih besar pada saat serangan *smurf attack* sebesar 120 paket dibandingkan dengan 20 paket, dan juga terdapat perbedaan penggunaan CPU pada serangan di

jaringan 1 dan 2 yakni antara serangan 1 dan 2 yang berada pada jaringan 1 dan serangan 3 dan 4 pada jaringan 2, serangan jaringan 1 yang memiliki *host up* sebanyak 21 *host up* pada jaringan tersebut menghasilkan penggunaan CPU yang lebih besar dibandingkan dengan serangan pada jaringan 2 yang hanya memiliki 5 host up pada jaringan tersebut.

TABLE IVV
NETWORK HISTORY PADA VM KORBAN

	Jumlah Paket Serangan	**Network History Sebelum Serangan**		**Network History Sesudah Serangan**	
		Total received	***Total sent***	***Total received***	***Total sent***
1	20 Paket	18.4 KiB	18.2 KiB	29.2 KiB	23.2 KiB
2	120 paket	126.6 KiB	17.8 KiB	162.7 KiB	20.4 KiB
3	20 paket	9.8 KiB	12.2 KiB	25.5 KiB	12.7 KiB
4	120 paket	3.9 KiB	8.4 KiB	9.8 KiB	12.2 KiB

Berdasarkan tabel IV di atas dapat dilihat bahwa terjadi kenaikan *terkait network history* pada setiap serangan *smurf attack* yang dilakukan pada jaringan, dapat dilihat terkait dengan *Total received* dan *Total sent* pada *network* terjadi peningkatan pada saat setiap serangan, akan tetapi tidak signifikan dapat dilihat pada saat serangan sebesar 20 dan 120 paket terjadi peningkatan pada *total received* dan *total sent* pada *network* tetapi dalam jumlah yang kecil.

IV. KESIMPULAN

Berdasarkan hasil yang didapat dari dampak yang dihasilkan oleh *smurf attack* terhadap komputer korban, didapati bahwa terjadi peningkatan, dalam hal penggunaan CPU yang cukup signifikan, hal ini menyimpulkan bahwa *smurf attack* memiliki dampak yang cukup besar terhadap penggunaan CPU dari korban serangan tersebut, yang menyebabkan komputer terasa menjadi lebih berat dari sebelumnya. Jumlah paket yang diterima oleh snort sebanyak 40 paket dan 240 paket jumlah ini sama dengan jumlah paket yang berhasil dianalisis oleh snort dan terindikasi sebagai sebuah serangan, yang berarti bahwa snort berhasil mendeteksi semua paket yang merupakan serangan *smurf attack* pada jaringan tersebut. Didapat bahwa terjadi peningkatan yang signifikan pada penggunaan CPU pada VM korban dalam hal ini VM dengan ip 192.168.88.114 dan 192.168.175.144,dari yang awalnya penggunaan pada CPU 1 hanya sebesar 27.4 %, 21.2 %, 18%, 15.4%, dan pada CPU 2 sebesar 27%, 24.3%, 20.7%, 13%, setelah dilakukan serangan meningkat menjadi 70.7%, 54.1%, 41%, 49.5% pada CPU1 dan 80%, 96.3%, 40.6%, 69.2% pada CPU2, hal ini membuktikan bahwa *Smurf Attack* memberikan pengaruh terhadap penggunaan CPU pada komputer korban.

REFERENSI

[1] Dar, M. H., & Harahap, S. Z. (2017). "Implementation of Snort Intrusion Detection System (IDS) on Computer Network Systems." Jurnal Informatika, 6(3), 14–23. [Online]. DOI10.36987/informatika.v6i3.1619
[2] Efendi, N. P. (2019). "Network Security System Using Snort." Jurnal Ilmu Informatika Dan Sistem Informasi, 1, 7.
[3] Dewi, E. K. (2017). "Analysis of Snort Logs Using Network Forensics." JIPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika), 2(2), 72–79. [Online]. DOI: 10.29100/jipi.v2i2.370
[4] Wijanarko, D. (2015). "Computer Network Security System Using SNORT." Jurnal Teknologi Informasi Dan Terapan, 02(01), 171–175.
[5] Anggrawan, A., Azhar, R., Triwijoyo, B. K., & Mayadi, M. (2021). "Developing Application in Anticipating DDoS Attacks on Server Computer Machines." MATRIK : Jurnal Manajemen, Teknik Informatika Dan Rekayasa Komputer, 20(2), 427–434.[Online]. DOI: 10.30812/matrik.v20i2.410.
[6] Jaw, E., & Wang, X. (2022). "A Novel Hybrid-Based Approach of Snort Automatic Rule Generator and Security Event Correlation (SARG-SEC)." PeerJ Computer Science, 8, e900. [Online]. Available: https://doi.org/10.7717/peerj-cs.900.
[7] Verma, N. (2022). "Detecting Network Threats using Snort Intrusion Detection System." Irjet, 9(1), 61–65.
[8] Saputra, B., Juli, S., Ismail, I., & Rizal, M. F. (2015). "Design and Implementation of NIDS (Network Intrusion Detection System) using Snort and BASE on FreeBSD 10." Journal/Conference Name, 1(2).
[9] Radhito, D., Widiyanto, A., Pujiarto, B., (2019). "Penetration on Computer Network at Biro Tik." Journal/Conference Name, 2(2).