Jurnal Integrasi
Vol. 11 No. 1, April 2019, 49-58
e-ISSN: 2548-9828

Article History
Received January 2019
Accepted April 2019

RANCANG BANGUN *OWNCLOUD* DAN MELINDUNGINYA TERHADAP SERANGAN *DDOS*

Molavi Arman[1*] dan Inayatullah[2*]

*AMIK MDP

Program Studi Manajemen Informatika

Jalan Rajawali, No.14, Palembang, 30113, Indonesia

E-mail: molavi.arman@mdp.ac.id[1] ; inayatullah@mdp.ac.id[2]

Abstrak

Layanan penyimpanan menggunakan Google Drive ataupun DropBox merupakan layanan yang digunakan secara cuma-cuma di internet, sebagai besar menggunakan teknologi *Cloud Computing*. Hal ini berarti layanan pada *Cloud Computing* dapat disediakan dengan cepat dan meminimalisir interaksi dengan penyedia layanan *vendor/provider Cloud Computing*. Tempat penyimpanan data yang sering menjadi kendala yang bersifat pribadi ataupun pekerjaan, seringkali menyulitkan dalam mengakses. Gangguan keamanan dari aktifitas jaringan seperti *Denial of Service* dan infeksi virus yang cukup beragam. Kesulitan berbagi data dengan inipun menjadi kendala terutama menyeting batas waktu dokumen yang dibagikan, dengan siapa saja dokumen dibagikan dan hak akses apa saja yang diberikan. Kendala seperti inilah yang dihadapi sehari-harinya dan cukup mengganggu fleksibelitas terutama dalam rutinitas pekerjaan, maka dibutuhkan metode *firewall iptables. Firewall iptables* yang telah dikonfigurasi akan menghasilkan berkurangnya gangguan tersebut sehingga layanan tetap bisa diakses walaupun serangan terus berlanjut.

Kata Kunci: *cloud computing, internet, denial of service, firewall, iptables*

Abstract

Storage services using google drive or dropbox are services used freely on the internet. Most of them use cloud computing technology. It means that the service on cloud computing could be provided quicky and it could minimize interaction with the vendor/provider. One of the constraints of data storage personally and professionally is the difficulty to access. There is also security disturbance from network activity such as denial of service and virus infection that varied. Difficulty to share data also becomes the main constraint especially in setting the limit of time of the document shared, with whom and what accessed is allowed. These are some of the daily constraints which disturb the flexibility in routine work. Therefore, firewall iptables method is needed. Firewall iptables that have been configured would minimize those disturbance so that the service could still be accessed even though the attacks still continue.

Keywords: *cloud computing, internet, denial of service, firewall, iptables*

1. PENDAHULUAN

Layanan penyimpanan menggunakan Google Drive ataupun DropBox merupakan layanan yang digunakan secara cuma-cuma di internet, sebagai besar menggunakan teknologi *Cloud Computing*. Beberapa definisi mengenai *Cloud Computing* oleh para ahli komputer. Secara umum dapat mengikuti salah satu definisi dan standarisasi yang diberikan mengenai *Cloud Computing*, salah satunya oleh NIST (*Natioanl Institute of Standard and Technology*). Didalam *draf*tnya yang berjudul The NIST *Definition of Cloud Computing* sebagai sebuah model yang memungkinkan adanya penggunaan sumber daya (*resource*) secara bersama-sama dan mudah, menyediakan jaringan akses di mana-mana, dapat dikonfigurasi, dan layanan yang digunakan sesuai keperluan (*on demand*). Hal ini berarti layanan pada *Cloud Computing* dapat disediakan dengan cepat dan meminimalisir interaksi dengan penyedia layanan

vendor/provider Cloud Computing [1]. Tempat penyimpanan data yang sering menjadi kendala yang bersifat pribadi ataupun pekerjaan, seringkali menyulitkan dalam mengakses. Gangguan keamanan dari aktifitas jaringan seperti *Denial of* Service dan infeksi *virus* yang cukup beragam, akibatnya merugikan pengguna akan kehilangan data dan kesulitan mengakses sumber layanan *cloud*. Kesulitan berbagi data dengan rekan kerja inipun menjadi kendala terutama menyeting batas waktu dokumen yang dibagikan, dengan siapa saja dokumen dibagikan dan hak akses apasaja yang diberikan. Kendala seperti inilah yang dihadapi sehari-harinya dan cukup mengganggu fleksibelitas terutama dalam rutinitas pekerjaan.

Ada beberapa penelitian terdahulu yang melakukan uji coba seperti dilakukan [2] membangun *private cloud* dari ancaman DoS (*Denial of Service*) diterapkan pada layanan tanpa virtualisasi. Pada [3] membangun *private cloud* menggunakan *hypervisor* OpenStack. Teknologi OpenStack adalah salah satu jenis *hypervisor* yang bisa digunakan untuk virtualisasi didalamnya dapat membangun sumber daya seperti *Virtual Machine* sebagai *server host*. Tipe serangan DDoS seperti dalam [4], ada beberapa tipe serangan yang mampu melumpuhkan layanan *private cloud*. Pada pengambilan nilai-nilai komponen dalam uji coba serangan DDoS terhadap layanan *cloud* [5] terdapat 4 komponen penilaian yaitu CPU*, memory*, *disk* dan *network ,* pada penelitian tersebut dilakukan pada layanan KVM dan VirtualBox .

Berdasarkan penjelasan diatas penelitian ini akan melengkapi yang belum diterapkan pada penelitian terdahulu dengan melakukan rancang bangun owncloud dan melindunginya terhadap serangan DDoS didalam lingkungan teknologi virtualisasi *hypervisor* yaitu *VMWare*.

2. TINJAUAN PUSTAKA

2.1 Jenis Serangan DDoS

Serangan DOS / DDoS penyerang mengirimkan paket langsung dari komputernya ke komputer situs korban tapi alamat sumber dari paket bisa dipalsukan. Ada banyak tools yang tersedia untuk memungkinkan jenis serangan ini untuk berbagai protokol termasuk ICMP, UDP dan TCP. Beberapa *tools* yang umum termasuk LOIC, bonesi, dan hping. Beberapa serangan DDoS umum dibahas di bawah ini :

2.1.1 UDP Flood Attack

Dalam serangan denggan tipe UDP *Flood Attack* mengirimkan sejumlah besar paket UDP ke sistem korban, karena ada padatan jaringan dan menipisnya bandwidth yang tersedia untuk permintaan layanan yang sah ke sistem korban [4].

Serangan UDP *Flood* memungkinkan saat penyerang mengirimkan paket UDP ke port acak pada sistem korban. Bila sistem korban menerima paket UDP, maka akan ditentukan aplikasi apa sedang menunggu di port tujuan. Ketika menyadari bahwa tidak ada aplikasi yang menunggu di pelabuhan, maka akan menghasilkan paket ICMP "*destination unreachable*" ke alamat sumber palsu. Jika cukup paket UDP dikirim ke pelabuhan korban, sistem akan turun. Dengan menggunakan alat DoS, alamat IP sumber dari paket serangan dapat dipalsukan dan dengan cara ini identitas sebenarnya dari korban sekunder dicegah dari keterpaparan dan paket balik dari sistem korban tidak dikirim kembali ke *zombie* [6] .

2.1.2 ICMP Flood Attack

Serangan ICMP *Flood* memanfaatkan Internet Control Protocol (ICMP), yang mana memungkinkan pengguna mengirim paket *echo* ke *host* jarak jauh untuk memeriksa apakah tetap *host* tetap ada. Lebih khusus lagi saat terjadi serangan DDoS ICMP Flood mengirimkan paket volume echo yang besar menuju *host* korban.[4]

2.1.3 SYN *Flood Attack*

Dalam serangan SYN Flood, korban dibanjiri koneksi setelah terbuka. Klien sistem dimulai dengan mengirimkan pesan SYN ke *server*. *Server* kemudian mengenali pesan SYN dengan mengirim pesan SYNC-ACK ke klien. Klien kemudian menyelesaikan koneksi dengan merespon ACK. Sambungan antara clien dan *server* kemudian terbuka, dan data spesifik layanan bias ditukar antara klien dan *server*.[4]

2.1.4 *Smurf Attack*

Dalam serangan "*smurf*", korban dibanjiri paket *Message Control Protocol* (ICMP) "echo-reply". Pada jaringan IP, sebuah paket dapat diarahkan ke mesin individual atau disiarkan ke keseluruhan jaringan. Ketika sebuah paket dikirim ke alamat broadcast IP dari sebuah mesin pada jaringan lokal, paket tersebut dikirimkan ke semua mesin pada jaringan tersebut. Dalam serangan "*smurf*" penyerang menggunakan paket permintaan *echo* ICMP yang diarahkan ke alamat *broadcast* IP dari lokasi *remote* yang menghasilkan serangan *denial-of-service*. Saat penyerang membuat paket ini, mereka tidak menggunakan alamat IP dari mesin mereka sendiri sebagai alamat sumbernya. Sebagai gantinya, mereka membuat paket palsu yang berisi alamat sumber palsu korban penyerang. Hasilnya adalah ketika semua mesin di perantara situs menanggapi permintaan *echo* ICMP, mereka mengirim balasan ke mesin korban. Korban terkena kemacetan jaringan yang berpotensi membuat jaringan tidak dapat digunakan. [4]

2.2. Denial of Service

Dalam [7], Pada prinsipnya *Denial of Service* merupakan penolakan layanan hanya dapat terjadi bila sekelompok pengguna layanan menjadi "istimewa" dari pada pengguna lainnya. Pengguna yang lebih istimewa adalah mereka yang bisa mendapatkan prioritas lebih tinggi atau akses yang lebih kuat ke layanan ini. Apabila pengguna yang lebih istimewa dapat menggunakan hak istimewa mereka secara individu, atau berkolusi, untuk mencegah pengguna

lain mengakses layanan yang ditentukan untuk jangka waktu melebihi waktu tunggu yang diinginkan atau *finite waiting time* (FWT) layanan tersebut. Hak istimewa tersebut dapat diperoleh dengan memanfaatkan berbagai kebijakan pembagian layanan yang diskriminatif dan mekanisme pembagian layanan yang cacat. Pengguna yang kurang beruntung menjadi tergantung pada perilaku pengguna yang lebih istimewa tersebut.

2.3 *iptables*

Dalam [4], Iptables adalah bagian dari proyek *Netfilter*. *Netfilter* adalah seperangkat Kernel Linux yang berkomunikasi dengan susunan jaringan. *Iptables* adalah perintah dan tabel struktur yang berisi set aturan yang mengontrol penyaringan paket. Iptables cukup kompleks, Iptables menyaring paket dengan bidang di header paket IP, TCP, UDP, dan ICMP. Sejumlah tindakan yang berbeda dapat diambil pada setiap paket, kunci iptables adalah kesederhanaan. Mulailah dengan konfigurasi minimum yang diperlukan untuk menyelesaikan pekerjaan, lalu tambahkan peraturan sesuai keinginan, tidak perlu membangun bangunan iptables yang luas dan faktanya, ini adalah ide yang buruk itu membuat sulit untuk mempertahankan dan memperburuk kinerja sistem.

2.4 *OwnCloud*

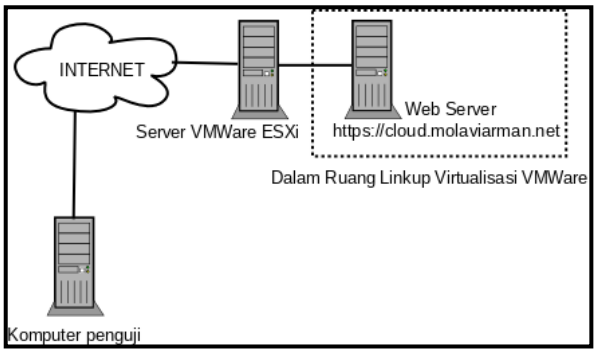
OwnCloud adalah aplikasi *open source* yang digunakan untuk berbagi perangkat lunak kepada semua orang dari orang-orang yang mengoperasikan edisi *ownCloud Server* gratis, kepada perusahaan besar dan penyedia layanan yang mengoperasikan langganan *Enterprise* sendiri. OwnCloud menyediakan solusi sinkronisasi dan berbagi *file* yang aman, aman, dan sesuai pada *server* yang anda kontrol. Aplikasi *OwnCloud* dapat berbagi satu atau lebih *file* dan *folder* di komputer anda, dan menyinkronkannya dengan *server cloud* anda sendiri. Tempatkan file di direktori bersama lokal anda, dan *file-file* itu segera disinkronkan ke *server* dan ke perangkat lain menggunakan aplikasi *Desktop Sync* Client, Android app, atau iOS sendiri.[8]

2.5 VMware ESXi: The Purpose-Built Bare Metal Hypervisor

VMware ESXi adalah *hypervisor* bare-metal yang dibangun langsung ke *server* fisik. Dengan akses langsung ke dan kontrol sumber daya yang mendasarinya, ESXi lebih efisien daripada arsitektur yang di-host dan dapat secara efektif mem-partisi perangkat keras untuk meningkatkan rasio konsolidasi dan memangkas biaya.[9]

2.6 Topologi Pengujian

Berikut adalah gambar topologi pengujian :



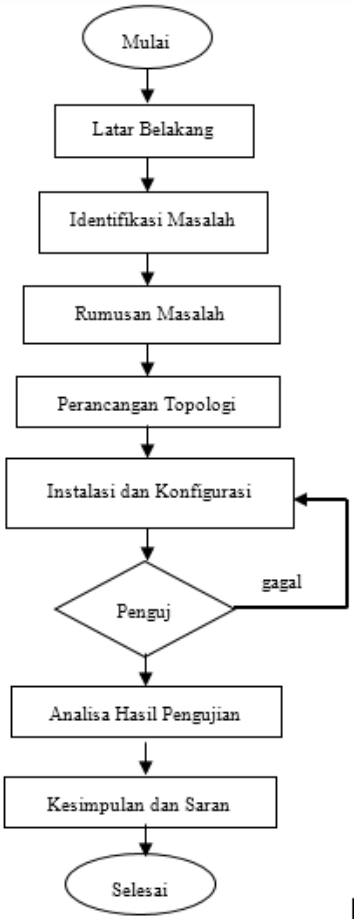
Gambar 1. Topologi Pengujian

Server VMWare ESXi sebagai hypervisor virtualisasi yang didalamnya terdapat *web server* https://cloud.molaviarman.net. Komputer penguji terhubung ke internet untuk melakukan pengujian dengan tools hping, bonesi dan LOIC.

3. METODOLOGI PENELITIAN

3.1. Diagram Alir Penelitian

Metodologi penelitian ini melalui beberapa tahapan yang direpresentasikan dalam suatu kerangka kerja penelitian / *framework.* Kerangka kerja penelitian secara sistematis menjelaskan langkah penelitian yang meliputi perancangan, pengujian,dan proses analisis.



Gambar 2. Diagram Alir Penelitian

Tahapan awal penelitian adalah merancang, membangun topologi *kemudian* melakukan instalasi dan konfigurasi *environment* yang dibutuhkan dalam membangun owncloud. Penjelasan diagram alir adalah sebagai berikut :

1. Menjelaskan latar belakang dengan menngumpulkan penelitian terdahulu, sehingga menjadi *landasan* dalam penentuan tema.
2. Melakukan indetifikasi masalah yang terjadi dalam virtualisasi dalam menghadapi serangan DDoS
3. Perancangan topologi yang dirancang untuk *server* owncloud diletakkan pada jaringan Internet tempat ISP yang menyediakan layanan *cloud*.
4. Instalasi dan konfigurasi *server owncloud* menggunakan sistem operasi slackware14.2 64 bit dan menggunakan memori sebesar 2GB, dengan kecepatan processor 1GHz.
5. Melakukan pengujian dengan menggunakan *tools* LOIC, hping2 dan Bonesi.
6. Pengambilan data diperoleh dari *ouput* pengujian meliputi indikator yang sudah dijelaskan.
7. Analisa pengujian memisahkan hasil sebelum konfigurasi dan setelah konfigurasi supaya menunjukkan adanya perbedaan terhadap *web server* owncloud sebelum dilindungi dan sesudah dilindungi.
8. Kesimpulan dari percobaan bahwa *web server* owncloud menunjukkan respon perubahan setelah dilakukan konfigurasi.

3.2. Objek Penelitian

Obyek penelitian ini adalah *server* yang baru dibangun yaitu sistem *Cloud Computing* dengan menguji coba performa *web server* dalam menghadapi serangan atau gangguan *flood* atau DDoS. Secara khusus obyek yang di uji adalah *web server Cloud Computing* yang akan di ukur kualitas layanan jaringannya. *Web server* owncloud akan melalui tahapan analisa untuk fungsi kerja dari *server* yang sudah dibangun dan analisa selanjutnya dengan melakukan uji eksperimental dengan melakukan serangan ke *web server cloud* yang kemudian mengambil nilai *output* data berupa CPU, *Disk*, *Memory* dan *Network*. [5]

4. PEMBAHASAN

4.1. Konfigurasi *Server*

Konfigurasi *web server* untuk meminimalisir serangan LOIC, hping dan bonesi diperlukan konfigurasi *firewall* menggunakan iptables. *Script* konfigurasi sebagai berikut :

Tabel I.

Firewall* Pada *Web Server

```
# Membatasi koneksi web server

iptables -t filter -I INPUT -p tcp --syn --dport 80 -m connlimit --connlimit-above 100 -j DROP
iptables -t filter -I INPUT -p tcp --syn --dport 443 -m connlimit --connlimit-above 100 -j DROP

## Membatasi ping

iptables -A INPUT -p icmp -m icmp --icmp-type address-mask-request -j DROP
iptables -A INPUT -p icmp -m icmp --icmp-type timestamp-request -j DROP
iptables -A INPUT -p icmp -m icmp --icmp-type 8 -m limit --limit 5/second -j ACCEPT

# Membatasi flooding dari Packet RST, Penolakan smurf attack

iptables -A INPUT -p tcp -m tcp --tcp-flags RST RST -m limit --limit 2/second --limit-burst 2 -j ACCEPT

# Melindungi dari portscans dengan pemblokiran 1 x 24 jam (3600 x 24 = 86400 Seconds)

iptables -A INPUT -m recent --name portscan --rcheck --seconds 86400 -j DROP
iptables -A FORWARD -m recent --name portscan --rcheck --seconds 86400 -j DROP

# Melepaskan serangan setelah 24 jam

iptables -A INPUT -m recent --name portscan --remove
iptables -A FORWARD -m recent --name portscan --remove

## Aturan-aturan ini menambahkan pemindai ke daftar portscan, dan log

iptables -A INPUT -p tcp -m tcp --dport 139 -m recent --name portscan --set -j LOG --log-prefix "portscan:"
iptables -A INPUT -p tcp -m tcp --dport 139 -m recent --name portscan --set -j DROP

iptables -A FORWARD -p tcp -m tcp --dport 139 -m recent --name portscan --set -j LOG --log-prefix "portscan:"
iptables -A FORWARD -p tcp -m tcp --dport 139 -m recent --name portscan --set -j DROP
```

Pada serangan LOIC semakin meningkat sehingga memakan sumber daya *memory* yang cukup tinggi, *web server* akan sangat terbebani walaupun serangan tersebut telah dihentikan. Menghadapi paska serangan tersebut diperlukan aplikasi monit untuk mengembalikan *memory* ke sediakala. Berikut konfigurasi monit :

Tabel II.

Konfigurasi Monit

```
if totalmem > 1024.0 MB for 5 cycles then restart
```

4.2. Pengujian

Web server https://cloud.molaviarman.net yang belum dikonfigurasi di uji dengan menggunakan *tools* LOIC, Bonesi dan hping dengan mengambil nilai CPU, *Disk, Memory,* dan *Nework*.[5] Berikut adalah hasil dari pengujian sebelum konfigurasi :

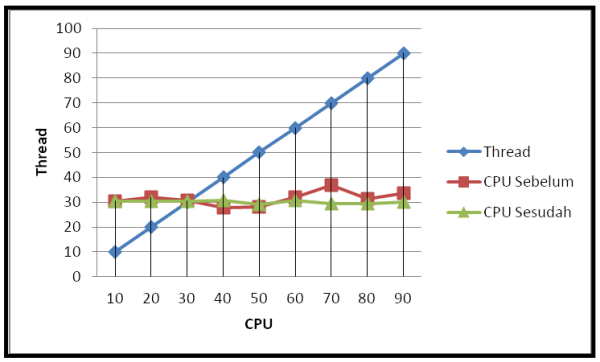
4.2.1.Serangan LOIC Sebelum dan Sesudah Konfigurasi Terhadap CPU

Perhatikan tabel III dan gambar 3 dibawah ini, hasil serangan LOIC yang menyerang *port https* 443 terhadap sumber daya CPU dengan tingkatan *thread* 10 sampai dengan 90.

Tabel III.

Dampak Serangan LOIC Terhadap CPU

Thread	**CPU Sebeleum**	**CPU Sesudah**
10	30.53	30.46
20	32.11	30.46
30	30.56	30.44
40	27.87	30.65
50	28.23	29.24
60	31.98	30.56
70	36.7	29.39
80	31.27	29.51
90	33.61	30.07



Gambar 3. Grafik Serangan LOIC terhadap CPU

Hasil dari pengujian terdapat selisih sebelum dan sesudah konfigurasi sebesar 1.04%. Selisih ini tidak mengganggu kestabilan *web server*.

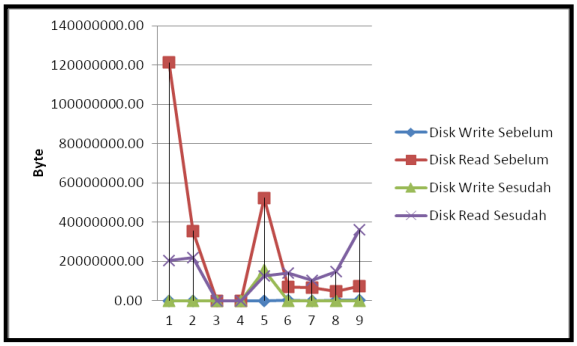
4.2.2 Serangan LOIC Sebelum dan Sesudah Konfigurasi Terhadap *Disk*

Perhatikan tabel 4 dan gambar 4 dibawah ini, hasil serangan LOIC yang menyerang *port https* 443 terhadap sumber daya *Disk* dengan tingkatan *thread* 10 sampai dengan 90.

Tabel IV.

Dampak Serangan LOIC Terhadap DISK

DISK write /byte (sebelum)	**DISK read /byte (sebelum)**	**DISK write /byte (sesudah)**	**DISK read /byte (sesudah)**
3.23	121271509.64	0.64	20448627.67
1.74	35475818.93	1.02	21802094.7
3.00	4.00	2	2
4.11	61368214.97	0.05	0
4512.63	52394107.7	15623001.95	12724915.92
309320.47	7111842.41	0.634622674	14235875.56
86917.64	6598393.735	1776.28	10521793.34
139814.61	4811245.11	0.629277098	14835909.73
225517.48	7505058.33	859.06	36102365.4



Gambar 4. Grafik dampak serangan LOIC terhadap Disk

Hasil dari pengujian *disk read* sebelum dan sesudah terdapat selisih 1%. Hasil dari pengujian *disk write* sebelum dan sesudah terdapat selisih 1.74% . Selisih ini tidak mengganggu kestabilan *web server*.

4.2.3 Serangan LOIC Sebelum dan Sesudah Konfigurasi Terhadap *Memory*

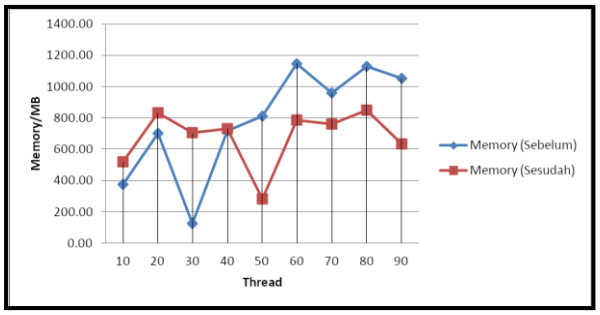
Perhatikan tabel V dan gambar 5 dibawah ini, hasil serangan LOIC yang menyerang *port https* 443 terhadap sumber daya *Memory* dengan tingkatan *thread* 10 sampai dengan 90.

Tabel V.

Dampak Serangan LOIC Terhadap *Memory*

Memory (sebelum)	**Memory (sesudah)**
376.31	517.87
701.66	832.5
128.33	704.53

718.49	732.64
810.43	283.74
1147.44	785.77
958.91	761.41
1128.04	849.06
1052.71	633.25



Gambar 5. Grafik dampak serangan LOIC terhadap Memory

Hasil dari pengujian *memory* yang digunakan pada saat serangan sesudah dan sebelum konfigurasi terdapat selisih sebesar 155.72%. Selisih yang cukup besar ini cukup mengganggu kinerja *web server*, untuk mencegah penggunaan *memory* yang berlebihan diperlukan aplikasi monit untuk memonitor penggunaan *memory* pada *daemon web server*.

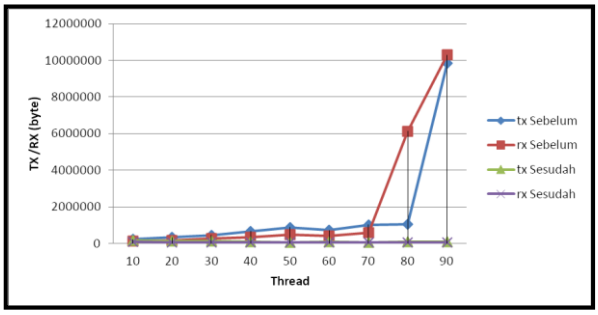
4.2.4 Serangan LOIC Sebelum dan Sesudah Konfigurasi Terhadap *Network*

Perhatikan tabel VI dan gambar 6 dibawah ini, hasil serangan LOIC yang menyerang *port https* 443 terhadap sumber daya *Network* dengan tingkatan *thread* 10 sampai dengan 90.

Tabel VI.

Dampak Serangan LOIC Terhadap TX/RX

tx Sebelum	rx Sebelum	tx Sesudah	rx Sesudah
215301.61	126436.28	142759.18	76157.49
326198.15	174190.43	119215.06	65286.55
451236.52	259842.47	109929.18	64236.88
663050.36	351993.90	82327.91	57088.23
870790.81	461207.47	66979.64	49222.70
720965.55	399811.85	71761.21	49434.42
1014657.68	586584.92	59769.96	49338.01
1051310.14	6131598.18	70243.50	60048.92
9833495.35	10290694.64	78897.33	64679.50



Gambar 6. Grafik dampak serangan LOIC terhadap tx dan rx

Hasil dari pengujian *network* tx sebelum konfigurasi dan sesudah konfigurasi terdapat selisih 18.85%. Hasil dari pengujian network rx sebelum konfigurasi dan sesudah konfigurasi terdapat selisih 35.07%. Selisih *traffic network* yang cukup signigikan ini cukup mempengaruhi kinerja *web server*, maka diperlukannya filter *iptables* sehingga bisa menghalau *traffic* yang tidak diperlukan

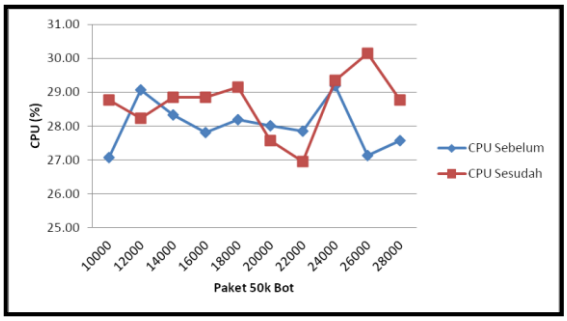
4.2.5 Serangan Bonesi Sebelum dan Sesudah Konfigurasi Terhadap CPU

Perhatikan tabel VII dan gambar 7 dibawah ini, hasil serangan Bonesi yang menyerang *port https* 443 terhadap sumber daya CPU dengan tingkatan 10000 sampai dengan 28000 paket.

Tabel VII.

Dampak Serangan Bonesi Terhadap CPU

Paket (50K bot)	CPU Sebelum	CPU Sesudah
10000	27.08	28.76
12000	29.07	28.23
14000	28.34	28.85
16000	27.81	28.85
18000	28.20	29.14
20000	28.01	27.58
22000	27.86	26.96
24000	29.19	29.35
26000	27.13	30.14
28000	27.58	28.76



Gambar 7. Grafik dampak serangan Bonesi terhadap CPU

Hasil dari pengujian terdapat selisih sebelum dan sesudah konfigurasi sebesar 0.98%. Selisih ini tidak mengganggu kestabilan *web server*.

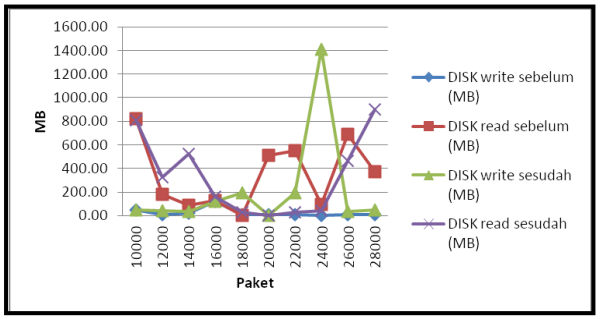
4.2.6 Serangan Bonesi Sebelum dan Sesudah Konfigurasi Terhadap *Disk*

Perhatikan tabel 8 dan gambar 8 dibawah ini, hasil serangan Bonesi yang menyerang *port https* 443 terhadap sumber daya CPU dengan tingkatan 10000 sampai dengan 28000 paket.

Tabel VIII.

Dampak Serangan Bonesi Terhadap *Disk*

***DISK write* sebelum (MB)**	***DISK read* sebelum (MB)**	***DISK write* sesudah (MB)**	***DISK read* sesudah (MB)**
46.30	821.58	49.65	807.28
11.11	180.69	40.92	327.57
20.14	86.29	32.50	524.79
118.62	128.47	122.01	159.25
5.77	2.79	193.74	29.68
8.25	508.46	0.00	0.00
8.02	548.55	191.07	28.07
2.23	92.74	1407.16	38.84
6.75	688.07	38.22	466.62
8.23	373.11	46.57	899.47



Gambar 8. Dampak serangan Bonesi terhadap Disk

Hasil dari pengujian *disk read* sebelum dan sesudah terdapat selisih 1.04%. Hasil dari pengujian *disk write* sebelum dan sesudah terdapat selisih 5.35% . Selisih ini tidak mengganggu kestabilan *web server*.

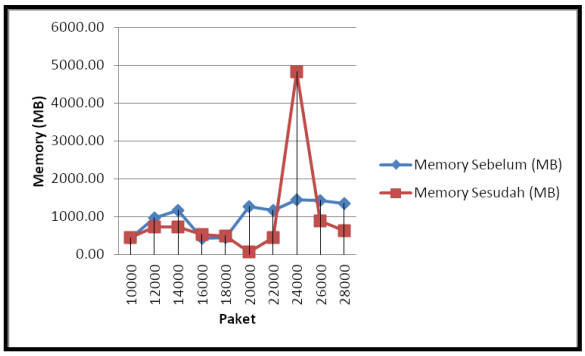
4.2.7 Serangan Bonesi Sebelum dan Sesudah Konfigurasi Terhadap *Memory*

Perhatikan tabel 9 dan gambar 9 dibawah ini, hasil serangan Bonesi yang menyerang *port https* 443 terhadap sumber daya *memory* dengan tingkatan 10000 sampai dengan 28000 paket.

Tabel IX.

Dampak serangan Bonesi terhadap *Memory*

***Memory* Sebelum (MB)**	***Memory* Sesudah (MB)**
416.76	439.23
968.42	727.90
1166.51	714.64
426.27	527.55
451.18	483.22
1268.09	56.61
1171.22	451.35
1443.54	4829.53
1423.61	883.75
1343.89	629.02



Gambar 9. Dampak serangan Bonesi terhadap *Memory*

Hasil dari pengujian *memory* yang digunakan pada saat serangan sesudah dan sebelum konfigurasi terdapat selisih sebesar 1.03%. Selisih ini cukup kecil dan tidak mengganggu kestabilan *server*, akan tetapi aplikasi monit tetap dijalankan untuk memonitor *daemon web server.*

4.2.8 Serangan Bonesi Sebelum dan Sesudah Konfigurasi Terhadap TX dan RX

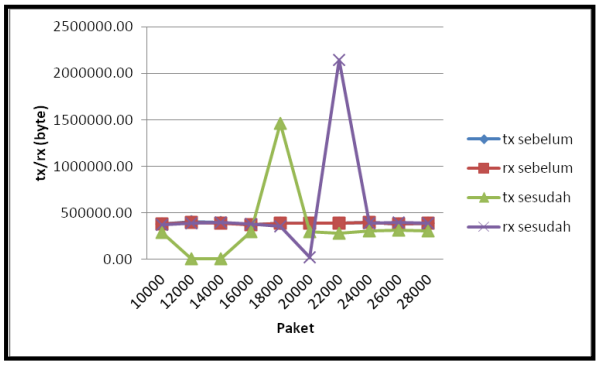
Perhatikan tabel X dan gambar 10 dibawah ini, hasil serangan Bonesi yang menyerang *port https* 443 terhadap sumber daya *Network* dengan tingkatan 10000 sampai dengan 28000 paket.

Table X.

Dampak serangan Bonesi terhadap TX dan RX

tx sebelum	**rx sebelum**	**tx sesudah**	**rx sesudah**
381112.31	379108.36	290943.33	374233.73
404069.98	393387.05	3958.24	385841.36
393190.96	387269.33	4224.12	394287.05
376064.23	373668.01	295154.59	377393.30

385529.24	388668.29	1463193.90	354877.33
387408.57	387408.57	295324.00	26468.07
389256.84	387606.70	280340.24	2141062.79
394739.55	396048.75	309667.19	389189.37
381385.89	382710.48	315023.88	396542.03
386178.46	384770.36	302993.47	390417.82



Gambar 10. Dampak serangan Bonesi terhadap TX dan RX

Hasil dari pengujian *network* tx sebelum konfigurasi dan sesudah konfigurasi terdapat selisih 1.09%. Hasil dari pengujian network rx sebelum konfigurasi dan sesudah konfigurasi terdapat selisih 0.73%. Selisih yang cukup kecil ini tidak mengganggu ke stabilan *traffic network* menuju *web server*.

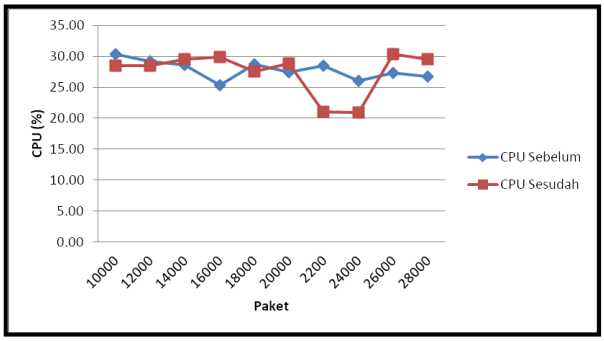
4.2.9 Serangan Hping Sebelum dan Sesudah Konfigurasi Terhadap CPU

Perhatikan table XI dan gambar 11 dibawah ini, hasil serangan Bonesi yang menyerang *port https* 443 terhadap sumber daya CPU dengan tingkatan 10000 sampai dengan 28000 paket.

Tabel XI.

Dampak Serangan Hping Terhadap CPU

CPU Sebelum	CPU Sesudah
30.40	28.45
29.17	28.48
28.61	29.58
25.36	29.94
28.69	27.58
27.42	28.85
28.48	21.03
26.08	20.91
27.33	30.38
26.73	29.57



Gambar 11. Grafik dampak serangan Hping terhadap CPU

Hasil pengujian menunjukkan terdapat selisih perbedaan terhadap sumber daya CPU sebesar 1.01%. Selisih tersebuh tidak berpengaruh terdapat sumber daya *web server* ketika terjadi serangan hping.

4.2.10 Serangan Hping Sebelum dan Sesudah Konfigurasi Terhadap *DISK*

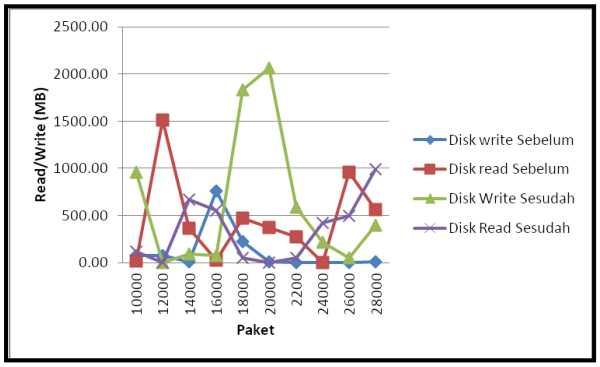
Perhatikan tabel XII dan gambar 12 dibawah ini, hasil serangan Bonesi yang menyerang *port https* 443 terhadap sumber daya *disk* dengan tingkatan 10000 sampai dengan 28000 paket.

Tabel XII.

Dampak Serangan Bonesi Terhadap Disk

Disk write Sebelum (MB)	*Disk read* Sebelum (MB)	*Disk Write* Sesudah (MB)	*Disk Read* Sesudah (MB)
72.77	18.07	960.40	119.12
79.80	1508.36	0.06	0.03
8.93	367.71	95.46	671.84
763.13	28.01	76.68	558.01
221.51	470.20	1836.63	54.00
9.66	373.49	2066.64	0.05
4.94	271.82	585.83	55.24
0.01	0.00	214.47	421.54
5.58	959.38	47.89	492.93
10.18	563.90	398.54	994.55

Hasil dari pengujian *disk read* sebelum dan sesudah terdapat selisih 0.19%. Hasil dari pengujian *disk write* sebelum dan sesudah *terdapat* selisih 1.35%. Selisih ini tidak mengganggu kestabilan *web server*.



Gambar 12 Grafik dampak serangan hping terhadap disk

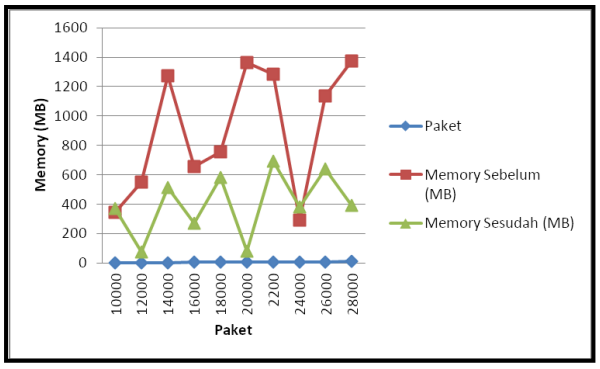
4.2.11 Serangan Hping Sebelum dan Sesudah Konfigurasi Terhadap *Memory*

Perhatikan tabel XIII dan gambar 13 dibawah ini, hasil serangan Bonesi yang menyerang *port https* 443 terhadap sumber daya memory dengan tingkatan 10000 sampai dengan 28000 paket.

Tabel XIII.

Dampak serangan Hping terhadap *Memory*

Memory Sebelum (MB)	*Memory* Sesudah (MB)
346.56	368.85
549.85	76.54
1272.38	513.57
658.27	271.36
754.39	580.32
1363.11	78.34
1281.78	693.87
289.56	381.25
1136.88	641.13
1371.14	393.56



Gambar 13. Dampak serangan Bonesi terhadap *Memory*

Hasil dari pengujian *memory* yang digunakan pada saat serangan sesudah dan sebelum konfigurasi terdapat selisih sebesar 2.25% dari 2GB memory pada *web server*. Selisih ini cukup kecil dan tidak mengganggu kestabilan *server*, akan tetapi aplikasi monit tetap dijalankan untuk memonitor daemon *web server,* ketika serangan telah selesai maka monit akan melakukan me-*refresh* terhadap konsumsi *memory* yang digunakan *web server.*

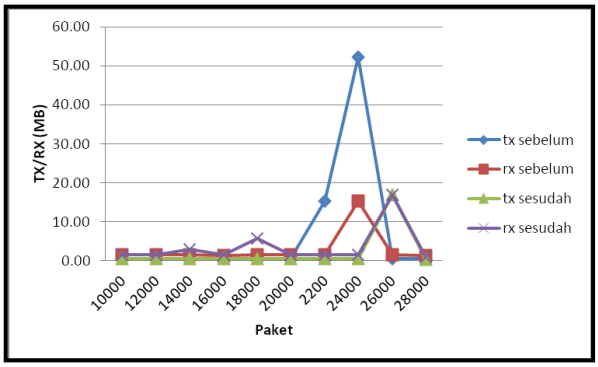
4.2.12 Serangan hping Sebelum dan Sesudah Konfigurasi Terhadap TX dan RX

Perhatikan tabel X dan gambar 10 dibawah ini, hasil serangan hping yang menyerang *port https* 443 terhadap sumber daya *Network* dengan tingkatan 10000 sampai dengan 28000 paket.

Table X1V.

Dampak serangan Bonesi terhadap TX dan RX (MB)

tx sebelum	rx sebelum	tx sesudah	rx sesudah
0.52	1.49	0.48	1.54
0.54	1.60	0.48	1.60
0.53	1.57	0.48	2.94
0.46	1.35	0.48	1.55
0.51	1.51	0.47	5.82
0.51	1.49	0.47	1.45
15.34	1.46	0.48	1.54
52.21	15.24	0.48	1.53
0.53	1.58	16.85	16.85
0.45	1.33	0.32	1.04
7.16	2.86	2.10	3.59



Gambar 14. dampak serangan hping terhadap sumber daya *network* TX *dan* RX

Hasil dari pengujian *network* tx sebelum konfigurasi dan sesudah konfigurasi terdapat selisih 2.5%. Hasil dari pengujian network rx sebelum konfigurasi dan sesudah konfigurasi terdapat selisih 0.8%. Selisih yang cukup kecil ini tidak mengganggu kestabilan *traffic network* menuju *web server.*

5 KESIMPULAN

Hasil pengujian terhadap 4 komponen yaitu kinerja CPU, *Disk*, *Memory* dan network TX/RX hal yang cukup menguras tenaga *web server* adalah konsumsi *memory* ketika dilakukan serangan DDoS hping terus meningkat walaupun serangan DDoS

tersebut telah selesai pada kondisi sebelum konfigurasi. Konsumsi memory yang terus meningkat hingga mendekat 95% dari besarnya *memory* yang dimiliki *web server* dengan kapasitas 2GB, sehingga sistem operasi mengalokasikan penyimpanan *memory* ke *memory swap* , kondisi ini cukup mengkhawatirkan. Setelah dilakukan konfigurasi terdapat perubahan yang cukup signifikan terhadap konsumsi *memory* terhadap daemon *web server*. Aplikasi monit sangat penting ketika serangan DDoS berakhir , dengan me-*refresh* terhadap konsumsi *memory* yang digunakan *web server* dengan menghasilkan konsumsi sumberdaya *memory* berjalan dengan normal.

6 REFERENSI

[1] Peter Meel, Timothy Grance, The NIST Definition of Cloud Computing, National Institute of Standar and Technology U.S. Departement of Commerce, Special Publication 800-145, Septemeber 2011.

[2] Johan Sharif, Membangun Private Cloud Computing dan Analisa Tehadap Serangan DoS, Study Kasus SMKN 6 Jakarta, IncomTech, Jurnal Telekomunikasi dan Komputer, vol.6, No.3, 2015.,

[3] Daniel Wilhenson Kuntani, Hendri Novianus Palit, Agustinus Noertjahyana, Implemetasi dan Evaluasi Cloud di Laboratorium Komputer, Jurnal Infra, Vol.3, No.2, 2015.

[4] Bahaa Qasim M. AL-Musawim, Mitigating DoS/ DDoS Attack Using IPtables, International Journal of Engineering & Technology IJET-IJENS, Vol. 12, No. 3, 2012.

[5] Pezhman Sheinidashtegol, Michael Galloway, Performance Impact of DDoS Attack on Three Virtual Machine Hypervisor, IEEE International Conference on Cloud Engineering, 4-7 April 2017.

[6] J. Markovic, J. Marti, and L. Reiher, A Mechanism CM Sigcomm Somputer Communication Reviewm Vol. 34, No.2 , pp. 39-53, 2004.

[7] Yu. C.-F, Gligor, V.D, A Specification and verification method for preventing denial of service, IEEE Trans, Software Eng, Vol. 16, Issue 6, pp. 581-592, June 1990.

[8] The onwCloud developers, ownCloud User Manual 9.1,https://doc.owncloud.org/server/9.1/ownCloud_User_Manual.pdf, diakses 4 April 2018.

[9] VMWare esxi, diakses https://www.vmware.com/products/esxi-and-esx.html, 4 April 2018.