Emulasi dan Analisis Keamanan Jaringan Virtual Data Center dengan Memanfaatkan sFlow dan OpenFlow untuk Mendeteksi dan Memitigasi Syn Flood

Ryanda Pratama HP[1], Sofia Naning Hertiana[2], R Rumani M[3]
Gedung Barung Ruang TE1.02.08, Universitas Telkom, Jl. Telekomunikasi No.1, Bandung 40257
[1,2,3] Program Studi Sistem Komputer, Fakultas Teknik Elektro – Universitas Telkom, Bandung
[1]ryandahp@gmail.com, [2]sofiananing@telkomuniversity.ac.id, [3]rumani@telkomuniversity.ac.id

Abstrak

Virtual data center merupakan salah satu teknologi yang bertujuan untuk mengurangi jumlah fisik data center dengan cara membuat mesin virtual dalam 1 server fisik. Dengan terciptanya virtual data center maka teknologi cloud computing dapat diimplementasikan. Salah satu platform cloud computing yang dapat mengelola resource dari data center adalah openstack. Ancaman dari teknologi cloud computing adalah dengan adanya ippublic yang sangat rentan dengan serangan dos (denial of service) diantaranya adalah synfloodattack. Telah terdapat sistem keamanan untuk mendeteksi synfloodattack, namun sistem keamanan yang ada masih diterapkan pada sisi server dan hanya diterapkan pada satu point pemasangan. Hal tersebut tentu akan menghabiskan resource dari sebuah server dan perangkat jaringan. Untuk mencegah hal tersebut, dibangun sebuah mesin virtual di dalam jaringan virtual data center yang berfungsi sebagai router dan firewall dengan memanfaatkan teknologi openvswitch yang mendukung teknologi SDN (Software Defined Network) dengan menggunakan protokol openflow dan sflow. Dengan memanfaatkan sflow-rt yang merupakan sflow collector sebagai pendeteksi serangan dan opendaylight yang merupakan openflow controller sebagai pemitigasi serangan maka serangan synfloodattack dapat dicegah sebelum memasuki mesin virtual data center dari openstack.

Kata kunci— *Virtual Data Center, Sflow, Openflow, Sflow-rt, Opendaylight*

Abstract

Virtual data center is one of the technologies that aim to reduce the number of physical data center by creating virtual machines within one physical server. With the creation of the virtual data center, then cloud computing technology will also able to implemented. One of the cloud computing platform that can manage the resources of the data center is openstack. Threats of cloud computing technology is the publicip, that highly vulnerable to DOS attacks (denial of service) which are synfloodattack. There has been a security system for detecting SYN flood attack, but the existing security system is implemented on the server side and only applied at the one point of installation. It is certainly going to spend resources on a server and network devices. To prevent that will be built a virtual machine within the virtual data center that will serve as a router and firewall by utilizing technology that supports technology openvswitch SDN (Software Defined Network) using the openflow protocol and sflow. By leveraging sflow-rt which is sflow collector as detecting attacks and opendaylight which is openflow controller as pemitigasi attack, then attacksynfloodattack can be prevented before it enters the virtual machine data center of openstack.

Keywords— *Virtual Data Center, Sflow, Openflow, Sflow-rt, Opendaylight*

1. PENDAHULUAN

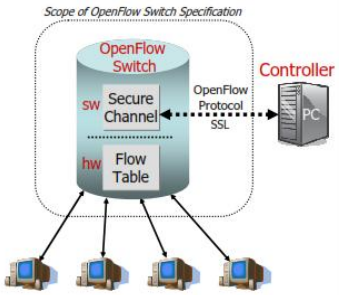
Teknologi *cloud computing* merupakan sebuah teknologi yang sangat bergantung pada *ippublic* atau internet. Kegunaan *ippublic* sendiri adalah penghubung antara *service* yang diberikan oleh *cloud computing* agar dapat diakses oleh *user* melalui jaringan internet. *Ippublic* dari sebuah *cloud computing* akan sangat rentan terhadap serangan *DOS* (*Denial of Service*s). Sudah banyak cara untuk melakukan

pencegahan dan pendeteksian terhadap serangan *DOS* ini, namun cara-cara yang telah ada masih memiliki kekurangan yaitu akan mengonsumsi *resource* dari *server*, karena diletakkan pada perangkat *server* tersebut. Untuk menangani hal tersebut dibutuhkan sistem keamanan yang diletakkan pada sisi perangkat jaringan dengan maksud untuk mengurangi *resource consumption* dari sebuah *server*. Salah satu cara untuk melakukan hal tersebut adalah dengan memanfaatkan teknologi *SDN (Software Defined Network)* yang menggunakan teknologi *sflow* dan *openflow*, dimana telah banyak penelitian sebelumnya [1,2], yang membuktikan bahwa metode tersebut dapat dijadikan sebagai sistem keamanan jaringan. Penelitian ini berdasarkan pada penelitian [3], dimana pada penelitian tersebut pemanfaatan *sflow* dan *openflow* sudah dapat mendeteksi dan memitigasi *syn flood attack*, namun pada penelitian tersebut target serangan dan penyerang masih dibuat di dalam *simulator mininet* dan menggunakan *floodlight controller*. Pada penelitian yang lain [4] telah dimanfaatkan *openvswitch* di dalam *hypervisor* untuk memitigasi *ddos attack* dengan menggunakan *opendaylight* sebagai *controller*, namun pada penelitian tersebut masih menggunakan tambahan 1 mesin *virtual* sebagai *virtual router*. Untuk mengurangi jumlah mesin *virtual* di dalam *hypervisor*, maka *openvswitch* akan dijadikan sebagai *firewall* dan *router* [5]. Pada penelitian ini dibuat sistem keamanan pada *virtual data center* dengan memanfaatkan *openvswitch* sebagai *firewall* dan *router* yang dihubungkan dengan *sflow-rt* dan *opendaylight controller* untuk mendeteksi dan memitigasi *syn flood attack*, selain serangan dilakukan dengan menggunakan komputer fisik sebagai penyerang, juga *resource* dari *instance* di dalam *virtual data center* tidak mengalami peningkatan.

2. METODE PENELITIAN

2.1. OpenFlow

Openflow [5] merupakan sebuah protokol yang sangat berkaitan dengan teknologi *SDN*. Jika pada jaringan tradisional sebuah *control plane* dan *data plane* menjadi satu dalam suatu perangkat jaringan, maka dengan adanya *openflow* kedua bagian tersebut akan dapat dipisahkan.

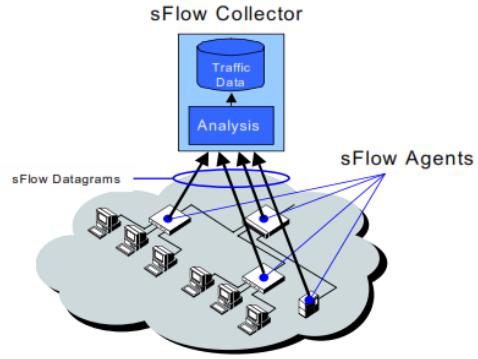


Gambar 1. OpenFlow Switch Specification [5]

Manfaat dari memisahkan kedua bagian tersebut adalah untuk mempermudah konfigurasi dari suatu perangkat jaringan, dikarenakan adanya sebuah *controller* yang dapat mengatur seluruh perangkat jaringan, dan terdapat banyak *controller* yang dapat digunakan antara lain *opendaylight* . Untuk menggunakan protokol *openflow* ini, dibutuhkan sebuah perangkat jaringan yang mendukung terhadap protokol *openflow* diantaranya adalah *openvswitch* .

2.2. sFlow

Sflow [6] merupakan sebuah teknologi yang fokus sebagai alat monitoring trafik dalam jaringan. *Sflow* dapat diterapkan pada semua *vendor* perangkat jaringan; salah satu perangkat jaringan yang dapat menggunakan fitur *sflow* adalah *openvswitch*. *Sflow* menggunakan teknologi *sampling* paket untuk memonitor jaringan. Dengan menggunakan *sampling* paket ini, perangkat jaringan tidak akan menghabiskan banyak *resources*. Terdapat 2 cara teknik *sampling* yang digunakan *sflow*, yaitu *sampling rate* dan *polling*.



Gambar 2. Teknologi sFlow [6]

Terdapat 3 bagian utama dalam teknologi *sflow* yaitu, *sflow agent*, *sflow collector* dan *sflow datagram*. *Sflow agent* merupakan *software* yang akan melakukan *sampling* paket di dalam *switch*. *Sflow datagram* merupakan paket yang sudah di cuplik (*sample*) yang akan dikirimkan menuju *sflow collector*; terdapat banyak *sflow collector* salah satunya adalah *sflow-RT* [6].

2.3. Virtual Data Center

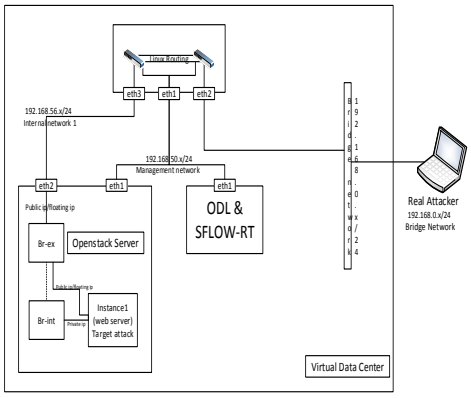
Virtual data center [6] merupakan sebuah teknologi yang dapat mengurangi jumlah fisik dari *data center* dengan membuat semua *resource* yang ada pada *data center* seperti , *network*, *memory*, *cpu* dan *storage* menjadi bentuk *virtual*. Teknologi ini merupakan bagian dari *IaaS*, yang merupakan *services* yang diberikan oleh *cloud computing*. Terdapat banyak cara untuk melakukan virtualisasi terhadap *data center* salah satunya adalah dengan menggunakan *openstack* [7].

2.4. Syn Flood Attack

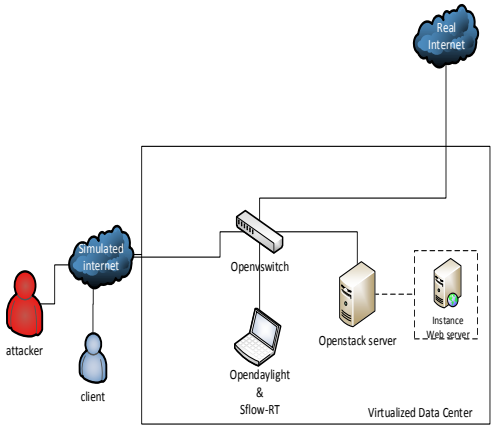
Syn flood attack [7] merupakan salah satu dari *dos attack*. Metode ini memanfaatkan *half-way open connection* dari *TCP*. Penyerang akan membanjiri *server* dengan paket *syn* sehingga *server* akan secara terus menerus mengirimkan kembali paket *syn-ack*. Efek dari metode ini adalah *server* tidak dapat melayani *request* yang lain dan *resource* dari *server* tersebut akan terus menerus meningkat.

2.5. Topologi Sistem

Untuk membuat sistem yang diinginkan, pertama-tama akan dibuat terlebih dahulu 3 mesin *virtual (openflow switch vm, opendaylight dan sflow-rt vm, dan openstack server vm)* di dalam *virtualbox hypervisor*. Selanjutnya dirancang sebuah *virtual data center* dengan menggunakan *openstack* [7]. Topologi yang dibuat adalah seperti yang terlihat dalam gambar 3 dan 4 berikut.

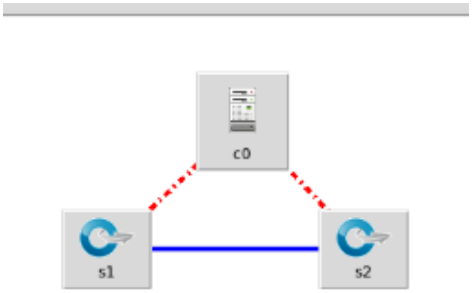


Gambar 3. Topologi Sistem Virtual Data Center



Gambar 4. Topologi Jaringan

Langkah selanjutnya adalah membuat *openflow switch* menggunakan *openvswitch*. *Openflow switch* ini nantinya menghubungkan *instance* di dalam *openstack* dengan jaringan luar; untuk melakukan hal tersebut akan dibuat 2 *ovs bridge*, s1 dan s2 menggunakan *miniedit* dan modifikasi *ip tables*.

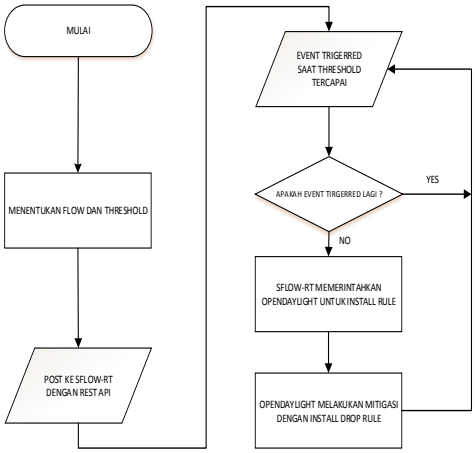


Gambar 5. Miniedit dengan 2 ovs

Setelah itu dilakukan konfigurasi terhadap setiap *ovs bridge* agar dapat menggunakan teknologi *sflow* dengan *command* berikut : ” sudo ovs-vsctl -- --id=@*sflow* create *sflow* agent=eth1 target=\"{*sflow-RT*-ip}:6343\" *sampling*=100 polling=20 -- -- set bridge s1*sflow=@sflow* ”

2.5. Sistem Kerja sFlow dan OpenFlow

Sistem kerja dari penelitian ini, mengikuti sistem kerja pada penelitian sebelumnya, dimana *sflow* bertindak sebagai deteksi serangan dan *openflow* bertindak sebagai mitigasi serangan. Secara garis besar sistem berjalan seperti *flowchart* di bawah ini.



Gambar 6. Flowchart System

Agar *sflow-rt* dan *opendaylight* dapat terhubung diperlukan perancangan *script* seperti penelitian sebelumnya [7] dengan melakukan modifikasi pada parameter yang digunakan. Agar *sflow-rt*dapat mendeteksi serangan *syn flood* diperlukan *filter* terhadap *tcp flags* dengan menambahkan *script* “*tcpflags*~…….1", pada bagian *filter*. Selain itu akan digunakan 800 *pps* sebagai *threshold*, dimana hal tersebut dilakukan setelah dilakukan beberapa macam percobaan.

Sistem mitigasi akan memerintahkan *opendaylight* untuk menginstal *openflow rule* yang telah ditentukan. *Openflow rule* yang digunakan adalah *drop rule*, *rule* ini akan melakukan *drop* paket dari *ip address* yang telah ditentukan.

Terdapat 3 skenario yang digunakan pada penelitian ini, dimana skenario 1 untuk menganalisis kondisi jaringan pada saat normal (tidak ada serangan), skenario 2 untuk menganalisis kondisi

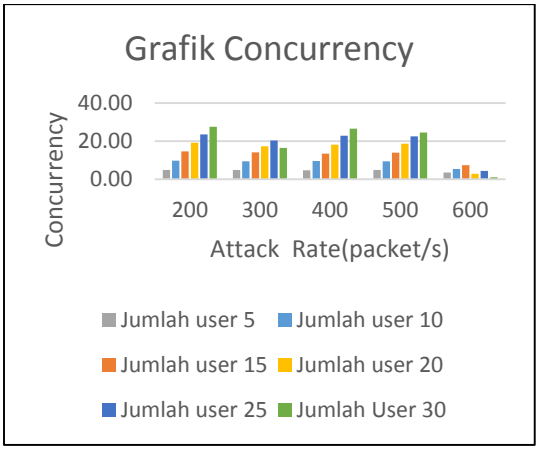
jaringan pada saat diserang, dan skenario 3 untuk menganalisis kondisi jaringan pada saat *block attack*.

Untuk kerja sistem sendiri pertama-tama *sflow* akan melakukan *sampling* terhadap seluruh paket yang melewati *openvswitch*. Pada sistem ini *sample rate* yang digunakan adalah 100. Seluruh paket yang sudah dicuplik (sampel), akan dikirim menuju *sflow-rt* untuk dianalisis dan ditampilkan dalam bentuk grafik. Dikarenakan telah dilakukan *filter* terhadap paket *syn*, maka grafik hanya menampilkan paket *syn*. Disaat serangan dilakukan maka jumlah paket yang melewati jaringan akan melewati *threshold* yang telah ditentukan sebelumnya, dan disaat *threshold* tercapai, maka *event* akan muncul. Disaat *event* muncul, maka aplikasi yang telah dibuat akan memerintahkan *opendaylight* untuk menginstal *openflow rule* yang sebelumnya telah ditentukan. *Openflow rule* disini berfungsi untuk melakukan *drop* seluruh paket dari *ip address* penyerang yang terdeteksi. Lalu *openflow rule* akan di *uninstall* disaat *block time* telah tercapai yaitu 20 detik; jika setelah 20 detik trafik masih melewati *threshold* maka *opendaylight* akan menginstal *openflow rule* kembali.

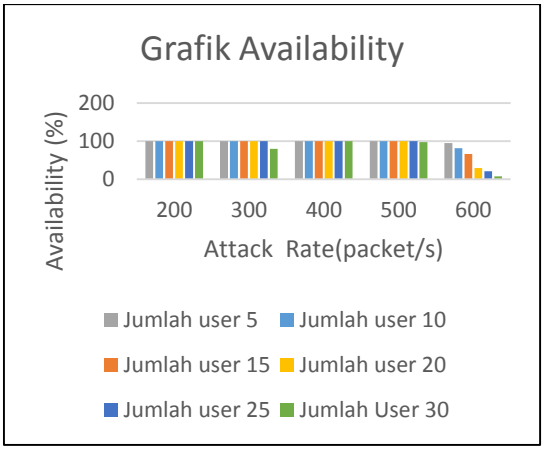
3. HASIL DAN PEMBAHASAN

3.1 Grafik sFlow-RT

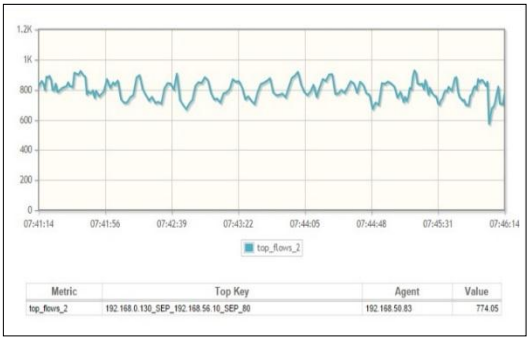
Percobaan ini dilakukan untuk menentukan nilai *threshold* yang akan digunakan. Percobaan ini dilakukan dengan megirimkan paket dengan *rate* yang berbeda-beda hingga *server* memiliki kondisi yang mulai *abnormal*. Untuk paket *rate* awal dilakukan percobaan dengan *packet rate* 200 hingga 600, dikarenakan saat *packet rate* 600 kondisi *server* sudah jauh berbeda dengan kondisi normal. Berikut hasil pengujian yang telah dilakukan.



Gambar 7. Grafik *Availability*



Gambar 8. Grafik *Concurrency*

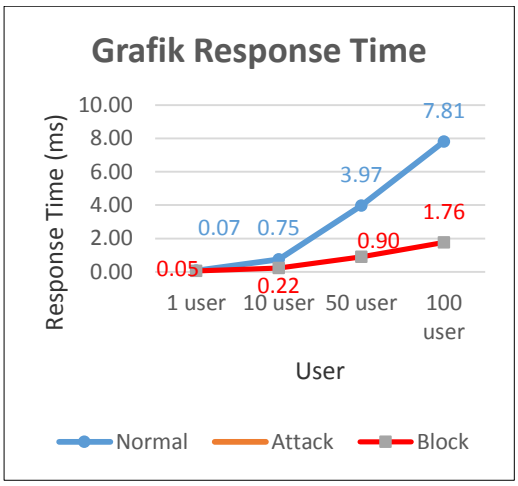


Gambar 9. Grafik sFlow-RT

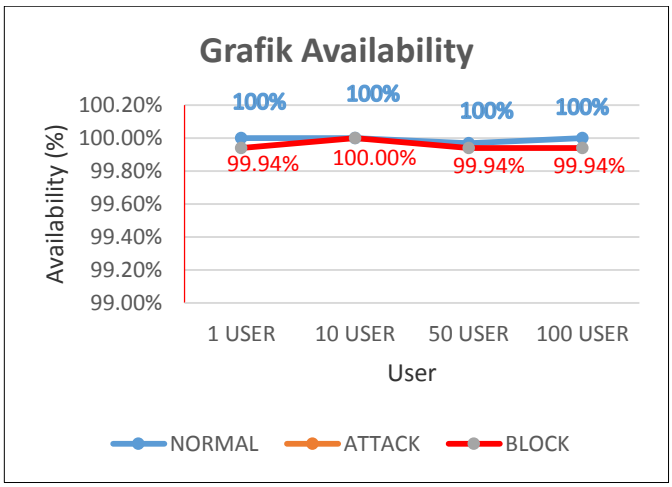
Berdasarkan grafik diatas dapat disimpulakn bahwa disaat *rate* mencapai 600 *pps*, kondisi *server* mulai mengalami penurunan, maka dari itu *threshold* yang akan digunakan berdasarkan pengamatan melalui *sflow-rt* adalah 800 *pps*.

3.2 Benchmark Webserver

Benchmark W*ebserver* merupakan salah satu cara untuk menguji kapabilitas dari *webserver* itu sendiri. Dalam melakukan *benchmark* untuk *webserver* yang telah dibuat, peneliti menggunakan *tool*s yang bernama *siege*. Dua hal yang dianggap penting dalam menguji *webserver* adalah *response time* dan *availability*. Pengujian dilakukan selama 1 menit dengan meningkatkan jumlah *user* mulai dari 1 *user*, 10 *user*, 50 *user*, dan 100 *user*. Berikut hasil pengujian yang telah dilakukan.



Gambar 10. Grafik *Response Time*

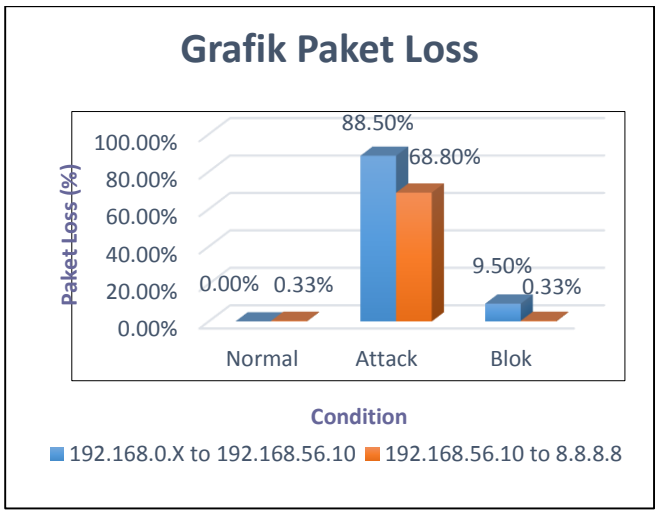


Gambar 11. Grafik *Availability*

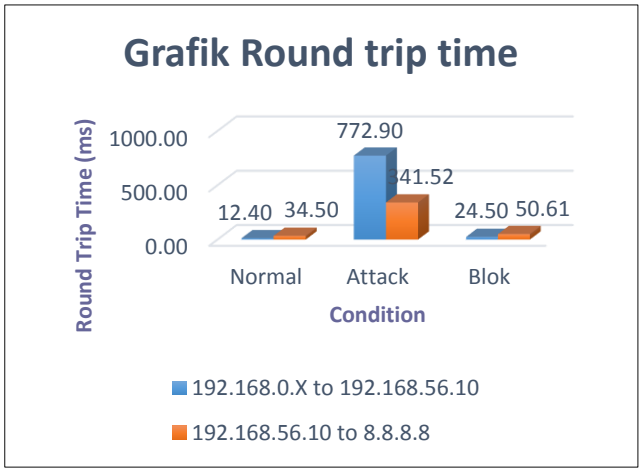
Dari kedua grafik di atas dapat disimpulkan bahwa sistem keamanan yang digunakan dapat meningkatkan *response time* sedangkan untuk tingkat *availability*nya sedikit berkurang dikarenakan sistem *block time* yang digunakan.

3.3 Packet Loss dan Round Trip Time

Pengujian ini dilakukan untuk melihat berapa banyak paket yang dapat melewati jaringan disaat kondisi sedang diserang. Diharapkan system keamanan yang telah dibuat mendapatkan nilai *packet loss* dengan nilai yang mendekati kondisi normal. Untuk melakukan pengujian *packet loss* dan *round trip time*, digunakan *command ping* dari *ip* 192.168.0.x menuju *ipserver* yaitu 192.168.56.10 dan dari *ip* 192.168.0.x menuju *ip google* 8.8.8.8; paket yang dikirimkan, sebanyak 100 paket. Berikut hasil dari pengujian *packet loss* yang didapatkan.



Gambar 12. Grafik *Packet Loss*

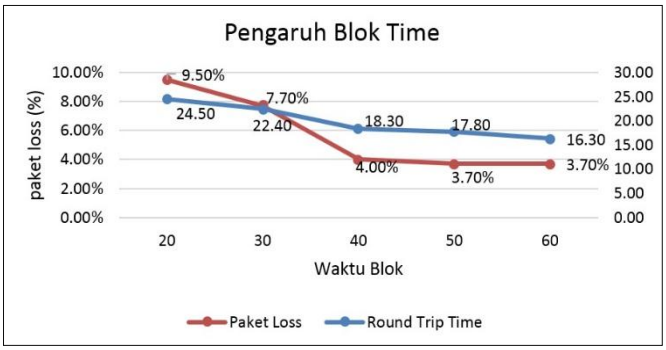


Gambar 13. Grafik *Round Trip Time*

Dari grafik di atas dapat dilihat bahwa sistem keamanan yang diterapkan dapat menghasilkan *packet loss* sebesar 9.5% dan 0.33% yang mendekati kondisi normal. Sistem keamanan yang digunakan tidak dapat menghasilkan *packet loss* dengan nilai 0% dikarenakan skema *block second* yang digunakan untuk melepas *block rule* yang telah diinstal. Hasil di atas menunjukkan bahwa sistem keamanan yang telah dibuat memiliki waktu *RTT* sebesar 24.5 ms dan 50.61 ms, dimana hal tersebut mendekati waktu *RTT* kondisi normal dan sangat jauh berbeda dengan waktu *RTT* kondisi serangan yang mencapai 772.9 ms. Kondisi tersebut menunjukkan bahwa kondisi jaringan atau trafik yang terjadi saat serangan berhasil di blok oleh sistem yang telah dibuat.

3.4. Pengaruh Block Time

Pada sistem yang telah dibuat terdapat sebuah skema *block time* dengan waktu yang telah ditentukan. *Block time* berfungsi sebagai berapa lama waktu *opendaylight* akan men-*drop* trafik yang berasal dari *ip* penyerang. Dalam percobaan ini dicoba waktu *block time* mulai dari 20 detik, 30 detik, 40 detik, 50 detik, dan 60 detik. Lalu dilihat bagaimana pengaruh dari *block time* tersebut terhadap *Round Trip Time* dan *Packet Loss* yang terjadi. Berdasarkan percobaan yang telah dilakukan, hasil yang didapat adalah seperti yang terlihat dalam grafik berikut ini.

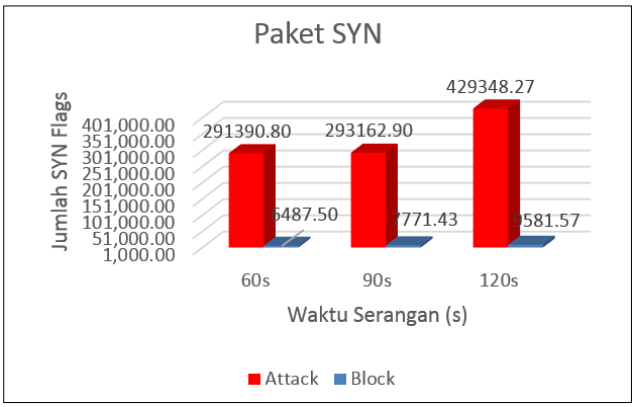


Gambar 14. Grafik pengaruh *block time*

Pengaruh *block time* dapat dilihat pada gambar di atas, dimana untuk *packet loss* dan *Round Trip Time* akan terus menurun disaat *block time* yang diterapkan semakin lama.

3.5. Jumlah Paket SYN

Percobaan ini dilakukan untuk melihat apakah paket syn telah diblok pada sisi *server*. Percobaan ini untuk membuktikan bahwa sistem keamanan dapat bekerja dengan baik, dengan hasil yang didapat, sebagai berikut :

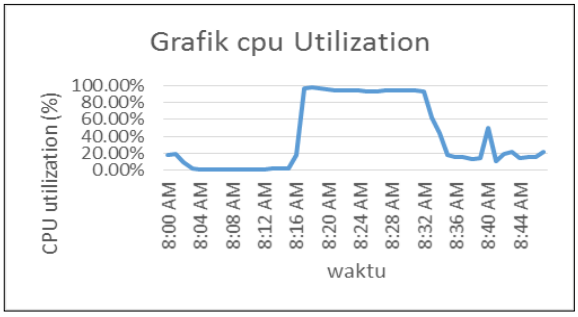


Gambar 15. Jumlah paket SYN

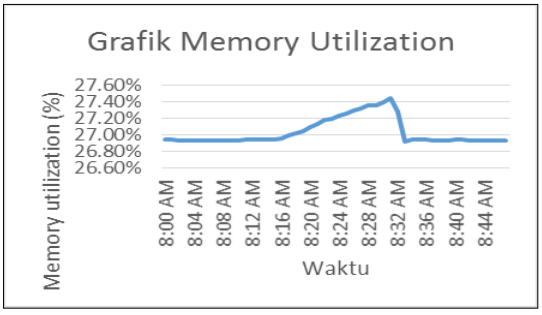
Berdasarkan grafik di atas paket syn yang masuk ke *server* berhasil berkurang secara signifikan; hal tersebut menandakan bahwa sistem keamanan dapat berjalan dengan baik.

3.6 Performansi Host

Performansi *host* sangat berpengaruh terhadap kondisi jaringan yang terjadi. Pada saat kondisi jaringan sangat padat, maka performansi *host* akan meningkat dan hal tersebut akan sangat mempengaruhi *resource* dari *server* yang digunakan. Jika performansi *host* dalam hal ini *cpu utilization* dan *memory utilization* meningkat maka hal tersebut dapat mengakibatkan *server* menjadi tidak berkerja atau menjadi sangat lambat. Untuk menguji sistem keamanan yang diterapkan maka dilihat seberapa berpengaruh sistem keamanan yang diterapkan terhadap performansi *host*. Berikut merupakan kondisi *cpu utilization* dan *memory utilization* pada saat kondisi *server* normal, sedang diserang dan sedang di blok.



Gambar 16. CPU Utilization



Gambar 17. Memory Utilization

4. KESIMPULAN

Berdasarkan hasil penelitian dari beberapa skenario yang telah dilaksanakan, maka dapat diambil beberapa kesimpulan sebagai berikut :

1. Teknologi sflow dan openflow terbukti, dapat digunakan sebagai sistem keamanan jaringan dengan melakukan pengaturan seluruh trafik yang melalui perangkat jaringan berdasarkan pemodifikasian flow table pada setiap openflow switch.
2. Pengimplementasian teknologi sflow dan openflow sangat bermanfaat dalam jaringan virtual data center, dikarenakan sangat bergantungnya jaringan virtual data center dengan sebuah virtual switch, dimana selain digunakan sebagai virtual router, virtual switch tersebut dapat juga diimplementasikan untuk sistem keamanan dengan menggunakan teknologi sflow dan openflow.
3. Berdasarkan aspek keamanan, sflow dapat digunakan sebagai alat untuk mendeteksi setiap trafik yang melalui perangkat jaringan sedangkan openflow dapat digunakan sebagai alat untuk

memitigasi serangan synflood dengan cara melakukan drop seluruh trafik yang berasal dari ip penyerang. Hal tersebut terbukti dari jumlah paket syn yang masuk ke server dapat berkurang secara signifikan yaitu menjadi 6487.5 dibandingkan saat terjadi serangan yaitu sebanyak 291390.8. Selain itu, untuk aspek keamanan dalam hal availability, sistem keamanan yang diterapkan mengalami sedikit penurunan tingkat availability yaitu dari 100% menjadi 99.94%.

4. Berdasarkan aspek performansi jaringan dalam hal ini packet loss dan Round trip time, sistem keamanan yang digunakan dapat mengurangi packet loss secara signifikan pada saat diserang yaitu dari 88.50% turun menjadi 9.50%, sedangkan untuk Round trip time sistem keamanan yang digunakan juga dapat mengurangi durasi Round trip time dari 772.90 ms menjadi 24.50 ms. Hal tersebut menandakan trafik yang awalnya sangat padat dengan paket syn dapat dikembalikan menjadi keadaan normal
5. Berdasarkan aspek performansi Host dalam hal ini CPU utilization dan memory utilization, sistem keamanan yang digunakan dapat mengembalikan performansi host ke keadaan normal; hal tersebut menandakan bahwa kondisi server berjalan normal dengan tidak melakukan pengiriman paket syn secara terus-menerus.

5. SARAN

Penelitian ini masih dapat dikembangkan agar menjadi sebuah teknologi yang lebih sempurna; untuk itu saran yang dapat direkomendasikan untuk penelitian selanjutnya, adalah sebagai berikut:

1. Agar dapat diterapkan pada versi *opendaylight* yang lainnya, dan digunakan pada *SDN controller* yang lainnya, seperti *floodlight, RYU, NOX, POX* dan lain-lain
2. Digunakan *physical openflow switch,* dengan lebih dari satu *switch,* lalu dianalisa apakah ada peningkatan pada performansi sistem keamanan.
3. Diterapkan sistem keamanan pada sisi *virtual switch openstack,* dan dilakukan serangan antar mesin *virtual* di dalam *openstack*.
4. Dilakukan penelitian terhadap *SDN controller*.
5. Diharapkan untuk dapat menerapkan sistem kemanan ini terhadap beberapa jenis serangan lain, seperti *UDP Attack, Ping Of Death, DNS Attack* dan lain-lain.

UCAPAN TERIMA KASIH

Penulis mengucapkan terimakasih kepada LPPM Universitas Telkom yang telah mendukung pendanaan yang diperlukan untuk terselenggaranya penelitian ini.

DAFTAR PUSTAKA

[1] M. Bolanowski, B. Twarog and R. Mlicki, 2015, Anomalies detection in computer networks with the use of SDN, *Measurement Automation Monitoring,* vol. 61, no. 09.

[2] M. Afaq, S. Rehman and W.-C. Song, 2015, Large Flows Detection, Marking, and Mitigation based on sFlow Standard in SDN, *Journal of Korea Multimedia Society,* vol. 18, no. 2, pp. 189-198.

[3] M. Nugraha, Utilizing OpenFlow and sFlow to Detect and Mitigate SYN Flooding Attack, 2014, *Journal of Korea Multimedia Society Vol. 17, No. 8, August 2014(pp. 988-994),* May.

[4] S. Bhat, G. Kanitkar, P. Kannan and S. Nair, Can SDN controller based NSCs help improve user experience of online games?, 2015, *Capstone Research Paper*.

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and T. Jonathan, 2008, Openflow: Enabling Innovation in Campus Network, *SIGCOMM Computation and Communication,* vol. 38, no. 2, pp. 69-74, March 14.

[6] P. Phaal, S. Panchen and N. McKee, 2001, InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks, *The Internet Society*.

[7] W. Eddy, 2007, TCP SYN Flooding Attacks and Common Mitigation, *The IETF trust*.