JURTEKSI (Jurnal Teknologi dan Sistem Informasi) ISSN 2407-1811 (Print)
Vol. VII No. 3, Agustus 2021, hlm. 245 - 250 ISSN 2550-0201 (Online)
DOI: https://doi.org/10.33330/jurteksi.v7i3.1199
Available online at http://jurnal.stmikroyal.ac.id/index.php/jurteksi

IMPLEMENTASI *FIREWALL* PADA DATA LINK LAYER MENGGUNAKAN ARSITEKTUR *SOFTWARE DEFINED NETWORK*

Herman Saputra[1*], Ruri Ashari Dalimunthe[1]
[1]Sistem Komputer, Sekolah Tinggi Manajemen Informatika dan Komputer Royal
Email: [*]hermansaputra4@gmail.com

Abstract: Software Defined Network has recently become a new architecture in computer network systems that offer convenience with a centralized control system. The Software Defined Network Architecture will separate the control plane and the data plane. Where the control plane will be placed on the controller that is stored and provides all the needs that exist in the network system. In Software Define Network, it still needs a lot of research, especially security, because it is the most important part in the network. One solution that can be used as security is a firewall. With the firewall, every data traffic will be monitored and also determine the entry and exit of data. In this research, a firewall will be implemented at the data link layer which will use a tree topology in the software define network and POX as the controller. The results of this study produce a firewall program with the python programming language which is placed on the POX controller. The code modification has successfully blocked, controlled, and monitored each packet based on the rules on the firewall.

Keywords: data link; firewall; pox controller; software defined network

Abstrak: Software Defined Network belakangan ini menjadi satu arsitektur baru pada sistem jaringan komputer yang menawarkan satu kemudahan dengan sistem kontrol terpusat. Pada arsitektur Software Defined Network akan memisahkan control plane dan data plane. Dimana control plane akan ditempatkan pada controller yang bertugas mengatur serta menyediakan segala keperluan yang ada pada sistem jaringan. Pada Software Define Network masih perlu dilakukannya banyak riset khususnya bidang keamanan, karena keamanan merupakan bagian terpenting dalam jaringan. Salah satu solusi yang dapat digunakan sebagai keamanan adalah firewall. Dengan adanya firewall maka setiap lalu lintas data akan dapat dimonitoring dan juga menentukan keluar masuknya suatu data. Pada penelitian ini akan diimplementasikan firewall pada data link layer yang akan menggunakan topologi tree pada software defined network dan POX sebagai Controllernya. Hasil pada penelitian ini menghasilkan satu program firewall dengan bahasa pemrograman python yang ditempatkan pada pox controller. Modifikasi kode telah berhasil untuk memblokir, mengontrol dan memonitoring setiap paket berdasarkan rule pada firewall.

Kata kunci: data link; firewall; pox controller; software defined network

JURTEKSI (Jurnal Teknologi dan Sistem Informasi) ISSN 2407-1811 (Print)
Vol. VII No. 3, Agustus 2021, hlm. 245 - 250 ISSN 2550-0201 (Online)
DOI: https://doi.org/10.33330/jurteksi.v7i3.1199
Available online at http://jurnal.stmikroyal.ac.id/index.php/jurteksi

PENDAHULUAN

Dimasa perkembangan teknologi yang sangat pesat, telah banyak aspek yang mengalami perubahan yang luar biasa besar, tidak terkecuali bidang arsitektur jaringan komputer. Hal ini disebabkan banyaknya rasa ketidaknyamanan atau ketidakpuasan yang menjadi dasar mengapa arsitektur jaringan perlu adanya perkembangan lebih lanjut [1]. Pada arsitektur konvensional, masih terdapat beberapa item yang masih memiliki kekurangan seperti *high operational coasts*, *diffucult to manage*, dan, *decentralization*, dan lain sebagainya [1],[2]. Salah satu teknologi baru yang hadir dalam mengatasi masalah tersebut adalah *Software Defined Network* (SDN) [3].

Software Defined Network (SDN) belakangan ini menjadi satu arsitektur baru pada sistem jaringan komputer yang menawarkan satu kemudahan dengan sistem kontrol terpusat [4]. Pada arsitektur SDN akan memisahkan *control plane* dan data *plane* [5]. Dimana *control plane* akan ditempatkan pada *controller* yang berperan mengatur serta memberikan kebutuhan yang diperlukan oleh sistem jaringan [6]. Pada SDN memiliki satu *Open Flow* akan memungkinkan akses akses secar langsung dan manipulasi *forwarding plane* dari sebuah perangkat jaringan misalnya *switch* dan *router* baik dalam bentuk fisik maupun virtual [6], [7].

Pada arsitektur SDN, *controller* memiliki peran yang sangat penting guna mengatur dan memonitoring setiap lalu lintas atau trrafik pada jaringan [8]. Salah satu peran dari *controller* lainnya adalah menangani masalah keamanan yang dapat terjadi sewaktu-waktu, seperti serangan *Denial of Service* (DoS) maupun *Distributed Denial of Service* (DDoS) [4], [9].

Sistem keamanan sangat diperlukan guna melindung semua perangkat yang terdapat pada jaringan, salah satunya kita dapat menerapkan *firewall* [10], [11]. Dimana *firewall* yang akan dipasang nantinya akan diisi dengan *rule* yang dapat digunakan untuk memblokir setiap gangguan berdasarkan pada *layer* 2 pada arsitektur OSI atau dalam bahasa sederhananya adalah berdasarkan MAC *Address* pada perangkat.

Penelitian mengenai SDN ini sudah banyak dilakukan didalam seperti pada [12] membahas tentang membangun *statefull firewall* pada arsitektur SDN, yang menghasilkan suatu sistem firewall yang dapat menangani berbagai bentuk serangan pada SDN seperti SYN *Flooding Attack*. Selanjutnya [4] melakukan penelitian mengenai analisis performa *Centralized Firewall* pada *Multi* Domain *Controller* di Arsitektur SDN. Dimana pada penelitian tersebut menarik kesimpulan dengan menerapkan *firewall* pada SDN *controller* dapat membuat controller menjadi lebih cepat dalam mendeteksi serangan yang dibuktikan dengan waktu rata-rata sebesar 6,042 *secound* dan penggunanan cpu sebesar 18,4% dalam menangani serangan didalam jaringan SDN. Selanjutnya pada [6] melakukan penelitian mengenai Metode *Deep Packet Inspection* yang diimplementasikan untuk Meningkatkan Keamanan Jaringan pada arsitektur SDN. Berdasarkan hasil pengujian yang dilakukan setelah SDN ditambahkan dengan *Deep Packet Inspection* mampu melakukan bloking paket yang mencurigakan di jaringan. Pada [7] melakukan pelitian mengenai perancangan firewall di transfort layer dan aplication layer pada *arsitektur software defined network*. Kemudian pada [11] melakukan penelitian mengenai pengaplikasian evolusi algoritma didalam

JURTEKSI (Jurnal Teknologi dan Sistem Informasi) ISSN 2407-1811 (Print)
Vol. VII No. 3, Agustus 2021, hlm. 245 - 250 ISSN 2550-0201 (Online)
DOI: https://doi.org/10.33330/jurteksi.v7i3.1199
Available online at http://jurnal.stmikroyal.ac.id/index.php/jurteksi

optimalisasi performa pada perangkat *embeded network firewall.* Pada penelitian selanjutnya [10], telah melakukan peneltian tentang pengaplikasian mekanisme *firewall* pada SDN. Pada penelitian ini mengusulkan mekanisme pemeriksaan lalu lintas jaringan ditingkat *transfort layer* dan *application layer*. Aplikasi yang dijalankan berupa program dengan bahasa pemroraman *python* pada *pox controller* dan *emulator mininet*.

Berdasarkan latar belakang serta penelitian sebelumnya yang telah banyak dilakukan maka penelitian ini bertujuan membangun suatu sistem keamanan *firewall* pada *layer data link* yang akan digunakan pada arsitektur SDN dengan *topology tree* dengan 1 *pox controller*, 4 *Switch* dan 9 buah *host*, yang nantinya akan dijalankan dengan emulator mininet. Untuk *firewall* akan dibangun dengan bahasa pemrograman *python* yang nantinya program *firewall* tersebut akan ditempatkan dan dijalankan pada *pox controller*.

METODE

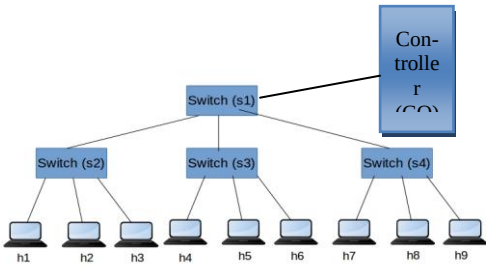
Kerangka kerja penelitian ialah tahapan sistematis yang dilakukan di dalam menuntaskan penelitian tentang Implementasi Firewall Pada Data *Link Layer* Menggunakan Arsitektur SDN. Adapun tahapan kerjanya terlihat pada gambar 1.

Gambar 1. Kerangka Kerja Penelitian

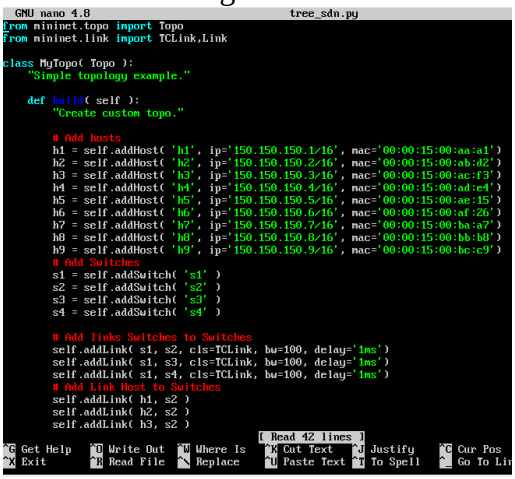
Dalam tahap perencanaan ini dilakukan pengumpulan data serta teori-teori dasar dari studi pustaka yang sesuai dengan SDN dan *Firewall.*

Pada tahap analisis ini ialah menganalisa akan kebutuhan penelitian. Hal ini diperlukan untuk menjadi acuan sistem yang dibangun. Aktifitas yang dilaksanan dalam menganalisis yakni memahami sistem yang akan dibangun dan metode pemecahannya.

Perancangan merupakan sebuah kegiatan merancang *topology tree* SDN. Untuk membuta topology pada arsitektur SDN memerlukan *emulator mininet* kemudian bahasa pemrogrman *python* unutk mendesain *topology*nya.



Gambar 2. *Topology Tree* SDN yang Digunakan

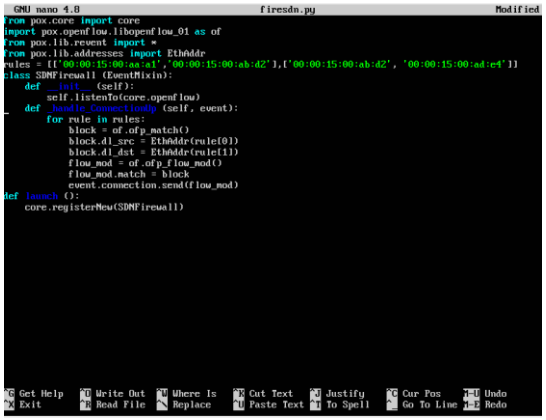


Gambar 3. *Source Code Topology Tree*

Pada gambar 2 merupakan bentuk dari *topology tree* pada SDN yang digunakan kemudaian pada gambar 3

JURTEKSI (Jurnal Teknologi dan Sistem Informasi) ISSN 2407-1811 (Print)
Vol. VII No. 3, Agustus 2021, hlm. 245 - 250 ISSN 2550-0201 (Online)
DOI: https://doi.org/10.33330/jurteksi.v7i3.1199
Available online at http://jurnal.stmikroyal.ac.id/index.php/jurteksi

merupakan tampilan dari *source code* untuk membuat *topology tree*. Untuk membuat *firewall* yang akan dijalankan bersamaan dengan *pox controller* juga dibuat dengan bahasa *python*, berikut tampilan *source codenya* pada gambar 4.



Gambar 4. *Source Code Firewall*

Pada gambar 4 merupakan implementasi dari *firewall rule* dalam bentuk bahasa *python* dimana rule yang dibuat adalah koneksi h1 dan h2 *bloked*. Kemudian h2 dan h4 *bloked*.

Pengujian sistem disini ialah tahapan untuk menguji sistem yang telah dibuat, apakah berjalan sesuai dengan tujuan yang ingin digapai dalam penelitian ini. Dalam pengujian ini nantinya akan dilakukan pengujian terhadap program untuk membuat topologi tree yang dijalankan dengan emulator mininet. Kemudian pengujian kedua adalah melakukan pengujian *firewall* yang dijalankan bersamaan dengan *pox controller*. Selanjutnya pada pengujian terakhir akan dilakukan pengujian *rule* pada *firewall* untuk memantau traffik dan memblokir beberapa perangkat pada level data *link layer*.

Pada tahap ini merupakan proses akhir yaitu menganalisa dan evaluasi hasil pengujian dan menarik kesimpulan dari penelitian ini.

HASIL DAN PEMBAHASAN

Software Defined Network (SDN) secara jelas bisa diartikan sebagai sebuah piranti lunak yang bisa mendefinisikan sebagai fungsi kontrol jaringan. Dimana untuk membangun SDN pada penelitian ini membutuhkan beberapa komponen diantaranya *python*, *pox controller* dan *Mininet*. Kemudia *software* khusus virtualisa yang mana disini menggunakan *Oracle VirtualBox*.

Tabel 1. MAC *Address* dan IP *Address* Pada Topologi SDN

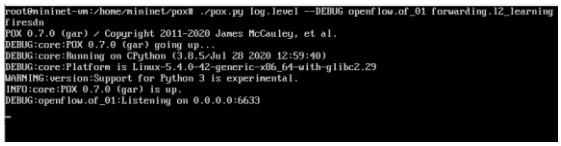
Perangkat	*Physical Address*	*Logical Address*
c0	08:00:27:d4:6d:65	10.10.12.253/24
h1	00:00:15:00:aa:a1	150.150.150.1/16
h2	00:00:15:00:ab:d2	150.150.150.2/16
h3	00:00:15:00:ac:f3	150.150.150.3/16
h4	00:00:15:00:ad:e4	150.150.150.4/16
h5	00:00:15:00:ae:15	150.150.150.5/16
h5	00:00:15:00:af:26	150.150.150.6/16
h7	00:00:15:00:ba:a7	150.150.150.7/16
h8	00:00:15:00:bb:b8	150.150.150.8/16
h9	00:00:15:00:bc:c9	150.150.150.9/16
s1,2,3 dan 4	*Default*	-

Pada tabel 1 merupakan tabel yang memperlihatkan MAC *Address* dan IP *Address* yang dikonfigurasikan pada *topology tree* yang digunakan.

Didalam penelitian *firewall* yang dipakai akan di letakkan pada *pox controller* sehingga nantinya akan berjalan bersamaan dengan *pox controller*. Untuk menjalankan *controller*

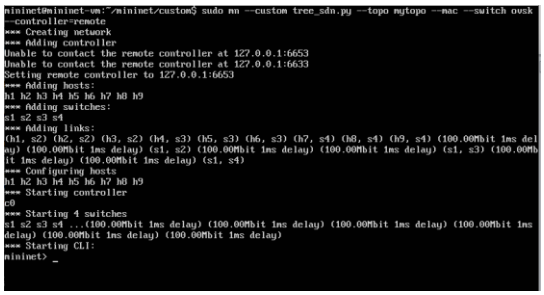
JURTEKSI (Jurnal Teknologi dan Sistem Informasi) ISSN 2407-1811 (Print)
Vol. VII No. 3, Agustus 2021, hlm. 245 - 250 ISSN 2550-0201 (Online)
DOI: https://doi.org/10.33330/jurteksi.v7i3.1199
Available online at http://jurnal.stmikroyal.ac.id/index.php/jurteksi

beserta *firewall* pada *mininet* kita terlebih dahulu harus mengakses *folder pox* pada *home directory*. Kemudian berpindah ke super *user* dengan perintah *sudo su*, lalu dapat mengetikkan perintah: *./pox.py log.level --DEBUG openflow.of_01 forwarding.l2_learning firesdn.*



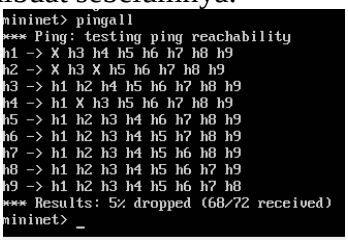
Gambar 5. *Start Pox Controller* dan *Firewall*

Pada gambar 5 memperlihatkan proses untuk menjalankan *pox controller* sebagai *controller* dengan fungsi *forwarding l2 learning.* Setelah dijalankan *pox controller* akan dalam mode *listening* pada *port* 6633 atau menunggu untuk meremote dari *topology* yang akan dilankan kemudian. Untuk menjalankan *topology* yang digunakan kita harus terlebih dahulu masuk *kedirectory* tempat penyimpanan *source code* untuk *topology*, disini kita perlu mengakses *directory mininet* kemudian sub *directory cus*tom. Dapat dilakukan dengan mengetikkan perintah *cd mininet/custom,* kemudian eksekusi perintah *sudo mn --custom tree_sdn.py --topo mytopo --mac --switch ovsk --controller=remote.* Tampilan lengkapnya terlihat di gambar 6.



Gambar 6. Menjalankan *Topology Tree* SDN

Seteleh *topology tree* dijalankan maka secara otomatis *pox controller* akan merespon dan membuat sambungan dengan mengkoneksikan ke *switch* utama. Setelah *pox controller* dan *topology* berjalan serta terhubung dengan baik. pengujian selanjutnya dilakukan untuk menguji dari *rule firewall* yang sudah dibuat sebelumnya.



Gambar 7. Pengujian *Rule Firewall*

Gambar 7 memperlihatkan pengujian dari *rule firewall* dimana h1 dan h2 *bloked* atau tidak dapat tersambung, kemudian h2 dan h4 juga disini masuk kedalam daftar *blakchlist* pada *firewall* berdasarkan rule yang sudah dibuat dengan memasukkan alamat *mac address* dari perangkat tersebut. Pada pengujian tersebut memperoleh data 5 % dari total koneksi di *dropped* atau 68/72 *data received.*

SIMPULAN

Bahasa pemrograman *python* sangat cocok diterapkan untuk pemrograman pada SDN. Penerapan *firewall* pada *data link layer* pada *topology tree* dengan arsitektur SDN telah berhasil untuk memblokir 5% dari total koneksi dan meneruskan koneksi antar *host* pada arsitektur SDN.

DAFTAR PUSTAKA

[1] L. O. Angreany Yasmin1) and 2), “Simulasi Software Defined

Network (SDN) untuk Manajemen Jaringan Data PT. Chevron Pasifik Indonesia," *Simulasi Softw. Defin. Netw. untuk Manaj. Jar. Data PT. Chevron Pasifik Indones.*, vol. 6, pp. 1–6, 2019.

[2] A. Tsuchiya, F. Fraile, I. Koshijima, A. Órtiz, and R. Poler, "Software defined networking firewall for industry 4.0 manufacturing systems," *J. Ind. Eng. Manag.*, vol. 11, no. 2, pp. 318–333, 2018, doi: 10.3926/jiem.2534.

[3] E. E. Thomas, H. Palit, and A. Noertjahyana, "Aplikasi Manajemen Jaringan Berbasis Software Defined Networking," *J. Infra Petra*, no. 031, 2018.

[4] R. P. Hidayat, R. Primananda, and E. R. Widasari, "Analisis Performa Centralized Firewall pada Multi Domain Controller di Arsitektur Software-Defined Networking (SDN)," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. Univ. Brawijaya*, vol. 2, no. 7, pp. 2698–2705, 2018.

[5] S. Zerkane, D. Espes, P. Le Parc, and F. Cuppens, "Software defined networking reactive stateful firewall," *IFIP Adv. Inf. Commun. Technol.*, vol. 471, pp. 119–132, 2016, doi: 10.1007/978-3-319-33630-5_9.

[6] D. P. Harja, A. Rakhmatsyah, and M. A. Nugroho, "Implementasi untuk Meningkatkan Keamanan Jaringan Menggunakan Deep Packet Inspection pada Software Defined Networks," *Indones. J. Comput.*, vol. 4, no. 1, p. 133, 2019, doi: 10.21108/indojc.2019.4.1.286.

[7] S. Badotra and J. Singh, *Creating Firewall in Transport Layer and Application Layer Using Software Defined Networking*, vol. 32. Springer Singapore, 2019.

[8] K. NUGROHO and D. P. SETYANUGROHO, "Analisis Kinerja RouteFlow pada Jaringan SDN (Software Defined Network) menggunakan Topologi Full-Mesh," *ELKOMIKA J. Tek. Energi Elektr. Tek. Telekomun. Tek. Elektron.*, vol. 7, no. 3, p. 585, 2019, doi: 10.26760/elkomika.v7i3.585.

[9] K. B. Virupakshar, M. Asundi, K. Channal, P. Shettar, S. Patil, and D. G. Narayan, "Distributed Denial of Service (DDoS) Attacks Detection System for OpenStack-based Private Cloud," *Procedia Comput. Sci.*, vol. 167, no. 2019, pp. 2297–2307, 2020, doi: 10.1016/j.procs.2020.03.282.

[10] F. N. Nife and Z. Kotulski, "Application-Aware Firewall Mechanism for Software Defined Networks," *J. Netw. Syst. Manag.*, vol. 28, no. 3, pp. 605–626, 2020, doi: 10.1007/s10922-020-09518-z.

[11] N. Lu and Y. Yang, "Application of evolutionary algorithm in performance optimization of embedded network firewall," *Microprocess. Microsyst.*, vol. 76, 2020, doi: 10.1016/j.micpro.2020.103087.

[12] K. Kaur and J. Singh, "Building Stateful Firewall Over Software Defined Networking Karamjeet," *Adv. Intell. Syst. Comput.*, vol. 434, no. January, pp. 179–186, 2016, doi: 10.1007/978-81-322-2752-6.