OPEN ACCESS
Indo-JC
Indonesia Journal of Computing
ISSN 2460-9056
socj.telkomuniversity.ac.id/indojc

Ind. Journal on Computing
Vol. 4, Issue. 1, March 2019. pp. 133-146
doi:10.21108/indojc.2019.4.1.286

Implementasi Metode Deep Packet Inspection untuk Meningkatkan Keamanan Jaringan pada Software Defined Networks

Danaswara Prawira Harja [#1], Andrian Rakhmatsyah [#2], Muhammad Arief Nugroho [#3]

 # Fakultas Informatika, Telkom University
Bandung, Indonesia

[1] danaswara@student.telkomuniversity.ac.id
[2] kangandrian@telkomuniversity.ac.id
[3] arief.nugroho@telkomuniversity.ac.id

Abstract

Today, Software Defined Network (SDN) has been globally recognized as a new technology for network architecture. But, there is still lack in security. Many studies use methods such as the Intrusion Prevention System (IPS) and Intrusion Detection System (IDS) to deal with social problems. But there is still a lack of security in terms of network performance. To solve the problem, can be used Deep Packet Inspection method (DPI) which make administrators can directly know what happens to traffic in real time. In this research, DPI will be implemented as security method and tested with Denial of Service (DoS) attack with Direct Attack. The results of testing on SDN networks that have been added DPI can block packets that are suspected during the attack are still canceled by the system which is shown by increased latency and bandwidth to be equal to 0 Mbits on network connections and false positive problems can be handle by splitting up level of attack that detected in network traffic.

Keywords: SDN, DPI, DoS attack, Direct Attack, performance

Abstrak

Saat ini, *Software Defined Network* (SDN) telah dikenal secara global sebagai teknologi baru untuk arsitektur jaringan. Tapi, masih banyak keraguan dalam segi keamanan. Banyak penelitian menggunakan metode keamanan seperti *Intrusion Prevention System* (IPS) dan *Intrusion Detection System* (IDS) untuk mengatasi masalah keamanan. Namun masih ada kekurangan dalam segi keamanan yang melibatkan performansi pada jaringan. Untuk memecahkan masalah tersebut, dapat digunakan metode *Deep Packet Inspection* (DPI) yang memungkinkan administrator memantau dan menganalisis secara mendalam apa yang terjadi pada lalu lintas secara *real time*. Dalam riset ini, diimplementasikan metode keamanan DPI pada arsitektur SDN yang diuji dengan teknik serangan *Denial of Service* (DoS) secara *Direct Attack*. Hasil dari pengujian pada jaringan SDN yang telah ditambahkan DPI dapat melakukan *blocking* paket yang dicurigai selama serangan masih terdeteksi oleh sistem pada arsitektur jaringan SDN yang ditandai dengan meningkatnya *latency* serta *bandwidth* menjadi sebesar 0 Mbits/s pada lalu lintas jaringan dan masalah *false positive* dapat diatasi dengan melakukan pembagian tingkatan serangan yang terdeteksi pada lalu lintas jaringan.

Kata Kunci: SDN, DPI, serangan DoS, *Direct Attack*, performansi

Received on Jan 2019. Accepted on March 2019

I. Pendahuluan

Software-Defined Networking (SDN) yang dipelopori oleh *Open Network Foundation* (ONF) merupakan arsitektur baru yang diperkenalkan sebagai jaringan masa depan [16]. Dalam SDN *control plane* dan *data plane* dipisah, diatur secara terpusat secara logika, dan infrastruktur jaringan dikendalikan dengan aplikasi. Pada jaringan berbasis SDN, *control plane* dan *data plane* terhubung dengan diatur oleh protokol yang dinamakan OpenFlow. OpenFlow memungkinkan akses langsung dan memanipulasi *forwarding plane* dari sebuah perangkat jaringan seperti *switch* dan *router*, baik fisik maupun *virtual* (berbasis *hypervisor*) [4] [5].

Ada ancaman serius yang dapat terjadi pada SDN, ancaman terbesarnya adalah *link* antara lapisan aplikasi dan jaringan yang terletak dibawahnya atau antara *Control Plane* dan *Data Plane* [2]. Beberapa penyelesaianya adalah dengan menggunakan metode yang ditawarkan oleh *Intrusion Detection System* (IDS) dan *Intrusion Prevention System* (IPS). Pada [17] [13] [10] [6] membahas mengenai IDS yang telah diintegrasikan pada arsitektur SDN, bahwa IDS memiliki performansi tinggi dalam hal ini *bandwidth* yang tinggi dan *latency* yang rendah karena hanya melakukan deteksi dan memberi peringatan ke administrator terhadap paket yang dicurigai, namun tanpa melakukan *blocking* yang membuat IDS rentan dalam segi keamanan.

Kemudian, pada [7] [15] telah dilakukan riset mengenai integrasi IPS pada arsitektur SDN yang mana hasil dari riset tersebut adalah IPS dalam hal keamanan lebih unggul daripada IDS dengan melakukan tindakan pencegahan berupa *blocking* paket yang dianggap oleh sistem berbahaya dan IPS melakukan *detection* serta *blocking* tidak seperti IDS yang hanya melakukan *detection*. Namun, IPS ini memiliki kekurangan dalam mengatasi *false positive* yang seringkali membuatnya memproses data yang sebenarnya tidak perlu diblokir sehingga menambah durasi blokir yang mengakibatkan sistem mengalami penurunan *bandwidth* dan *latency* yang tinggi, dan IPS memiliki durasi blokir yang lama untuk memproses suatu paket karena melakukan proses *detection* dan *prevention* paket yang berjalan di jaringan.

Untuk menyelesaikan masalah tersebut, salah satunya dengan menggunakan *Deep Packet Inspection* (DPI). Penjelasan mengenai DPI ini dijelaskan dalam artikel yang dikeluarkan oleh Radisys [8] bahwa DPI mampu mengumpulkan informasi secara mendalam dengan memanfaatkan modul komunikasi dari *layer* dua atau *presentation* hingga *layer* tujuh atau *physical*. Pada [3] dengan menggunakan metode DPI, SDN menjadi lebih aman karena dapat melakukan *monitoring*, *analyzing* dan *controlling traffic* dari *layer* 2 atau *data link* hingga *layer* 7 atau *application*. DPI juga mampu mengatasi kekurangan IPS dalam hal waktu proses pemblokiran dengan menitikberatkan pada *pattern matching* yang terjadi akibat adanya *false positive*.

II. Studi Terkait

Pada [12] membahas tentang bagaimana cara untuk mencapai jaringan yang baik dengan menggunakan *Software-Defined Networking* (SDN) dan menjawab tantangan mengenai performansi jaringan, skalabilitas, keamanan dengan arahan solusi yang baik, karena SDN teknologi jaringan yang efisien dan mampu untuk mendukung perubahan jaringan yang dinamis. Pada [4] membahas mengenai analisis keamanan pada arsitektur SDN khususnya pada protokol inti dari SDN, yaitu OpenFlow dengan berbagai kemungkinan ancaman yang terjadi dan juga diberikan rekomendasi langkah-langkah perbaikannya secara rinci dengan persyaratan kemanan yang relevan. Pada [2] mengeksplorasi ancaman utama yang ditimbulkan pada lingkungan arsitektur SDN dan secara komparatif menganalisis beberapa mekanisme yang diusulkan sebagai mitigasi terhadap ancamannya, dimana hasilnya menyatakan bahwa kelemahan terbesar dari keamanan SDN adalah pada link antara lapisan aplikasi dan jaringan yang terletak dibawahnya.

Sementara itu, pada [17] dibahas mengenai pembangunan teknologi keamanan IDS (*Intrussion Detection System*) yang memiliki kemampuan untuk mencari lokasi dan identifikasi aktifitas berbahaya pada lalu lintas jaringan secara *real time* pada arsitektur jaringan SDN agar dapat memberikan administrator *visibility* dan *realibility* untuk mengontrol serta memonitor lalu lintas jaringan, sedangkan pada [13] dibahas mengenai konsep baru pada protokol OpenFlow dengan menanamkan IDS) didalamnya agar lebih aman, dimana pada OpenFlow *switch* ada dua buah tabel baru yaitu IDS IP (*Internet Protocol*) *information* yang akan mencari data mengenai *source* IP dan IDS *Database* akan digunakan saat *source* IP tidak ditemukan pada data.

Pembahasan mengenai IPS (*Intrussion Prevention System*) dijelaskan pada [7], mengenai perancangan IPS *adaptive* dengan memanfaatkan logika *fuzzy* untuk memutuskan durasi blokir berdasarkan variabel frekuensi dan jenis serangan yang ada dan diterapkan sebagai solusi mengenai IPS yang bermasalah dalam waktu ketika melakukan deteksi dan *blocking* serangan agar sesuai dengan frekuensi dan jenis serangannya dimana didapatkan hasil yang baik dengan terdapat penambahan 0,228 milidetik sebagai *execute time* yang dibutuhkan algoritma *fuzzy* dalam sekali proses. Lalu pada [15] dibahas mengenai pembangunan SDNIPS dan melakukan perbandingan antara dua metode keamanan, yaitu IPS (*Intrussion Prevention System*) pada arsitektur tradisional dan SDNIPS pada arsitektur SDN (*Software Defined Networks*) dengan melakukan dua tipe serangan yaitu DDoS (*Distributed Denial of Service*) dengan ICMP (*Internet Control Message Protocol*) *flood* untuk melihat performansinya, dimana hasilnya saat paket serangan yang dikirimkan sebanyak 15000/detik IPS tradisional hanya bisa men-*generate* 13.72% saja berbeda dengan SDNIPS yang mampu menangani serangan hingga pada 30000 paket/detik SDNIPS menurun performansi untuk men-*generate* paket pada lalu lintas jaringan.

Pada [8] dibahas mengenai *Deep Packet Inspection* (DPI) mulai dari latar belakang adanya DPI, definisi, pendekatan mengenai DPI, fitur yang dimiliki oleh DPI, Fungsi dari DPI, hingga kasus dan penyelesaiannya menggunakan DPI. Dengan masih berkembangnya SDN yang masih relatif baru di dunia jaringan, maka masih banyak tantangan yang menjadi riset para pengembang, salah satunya adalah kemanan. Salah satu solusinya adalah dengan penggunaan DPI di SDN, karena DPI menyediakan layanan untuk semua fungsi yang relevan dimana pada [3] membahas mengenai usulan arsitektur SDN yang baru diimplementasikan dengan modul DPI diparalelkan, karena dengan melakukan hal tersebut akan memberikan informasi lebih detail mengenai arus lalu lintas dalam jaringan yang sedang berjalan sehingga jaringan lebih aman serta dengan mem-paralelkan algoritma DPI di SDN secara efektif mengurangi waktu untuk mendeteksi paket dan mengirim *flow tables*.

Pada [9] membahas secara detail mengenai DPI dimulai dari definisi dan fungsinya, serta *leveling* dari *packet inspection*nya, dan membahas mengenai isu yang ada pada DPI. Pada [11] membahas mengenai cara mengaktifkan modul DPI untuk *filtering traffic* dengan memonitor paket dan mengidentifikasi *string* dari data yang didapat menggunakan tools bernama Snort.

III. Perancangan Sistem

Pada bab sistem yang dibangun ini dibagi menjadi empat bagian, yaitu Gambaran Umum Sistem, Rancangan Sistem, Rancangan Topologi, dan Skenario Pengujian.

A. Gambaran Umum Sistem

Sistem yang dibangun pada penelitian ini merupakan sistem keamanan yang diimplementasikan pada arsitektur jaringan *Software Defined Network* (SDN), dimana SDN merupakan arsitektur jaringan dengan logika terpusat. *Tools* keamanan yang digunakan yaitu ntopng karena memiliki tingkat keakuratan tinggi dalam mendeteksi serangan [14], dengan memanfaatkan modul *Deep Packet Inspection* (DPI) yang dimiliki ntopng untuk kemudian melakukan *monitoring*, *analyzing*, dan *controlling traffic* dari *layer* 2 atau *presentation* hingga *layer* 7 atau *physical*. ntopng yang menggunakan modul DPI ini akan ditempatkan di *control plane* yang mana menjadi pusat pengaturan dari arsitektur SDN.

SDN berjalan dengan Floodlight yang bertindak sebagai *controller* atas jaringan yang terintegrasi di SDN. Dengan kata lain, seluruh paket yang berada dalam jaringan terlebih dahulu diproses oleh Floodlight, baru kemudian ntopng bertindak untuk menganalisa paket tersebut berdasarkan aturan yang telah dibuat. Jika paket terindikasi dalam bahaya, maka paket tersebut diblokir dan kemudian ditindak lanjuti, begitupun sebaliknya jika paket tidak berbahaya, maka paket diteruskan oleh *controller* ke tujuan.

Sistem keamanan jaringan DPI diimplementasikan di SDN pada bagian *control plane*, dengan tujuan untuk mengurangi kinerja dari DPI agar tidak melakukan analisa semua paket yang melintas dalam jaringan. Algoritma yang digunakan oleh DPI untuk melakukan *pattern matching* adalah Aho-Corasick, yang mana algoritma ini berada pada *tool* ntopng agar performa jaringan saat melakukan analisa pada paket dapat berjalan dengan efesien.

B. Alur Kerja Sistem

Gambaran secara umum mengenai alur kerja sistem ini telah dijelaskan pada bagian A bab Perancangan Sistem sedangkan pada bagian B ini dijelaskan lebih detail lagi tentang sistem yang diimplementasikan.

Modul DPI yang diaktifkan dengan *tool* ntopng ini diletakkan di *control plane* pada arsitektur jaringan SDN yang mana akan memungkinkan DPI agar tidak melakukan analisa terhadap semua paket yang melintas pada jaringan, karena akan ditangani langsung oleh protokol OpenFlow pada *controller* sebelum masuk ke DPI.

Proses yang akan dilakukan DPI dapat dilihat pada Gambar 1, dimana terdiri dari empat tahapan, yaitu *getting original packet*, untuk mendapatkan paket yang sebelumnya ada di Floodlight untuk di analisa. Kemudian *flow pretreatment*, untuk mengetahui aliran data dari paket sebelum diproses. Lalu, *pattern matching*, yang merupakan proses paling menentukan apakah performa dari DPI baik ataupun buruk performansi waktu, dalam melakukan deteksi paket berdasarkan *pattern* yang ada di paket. Selanjutnya adalah *writing flow table*, untuk menentukan paket diteruskan atau di blokir.

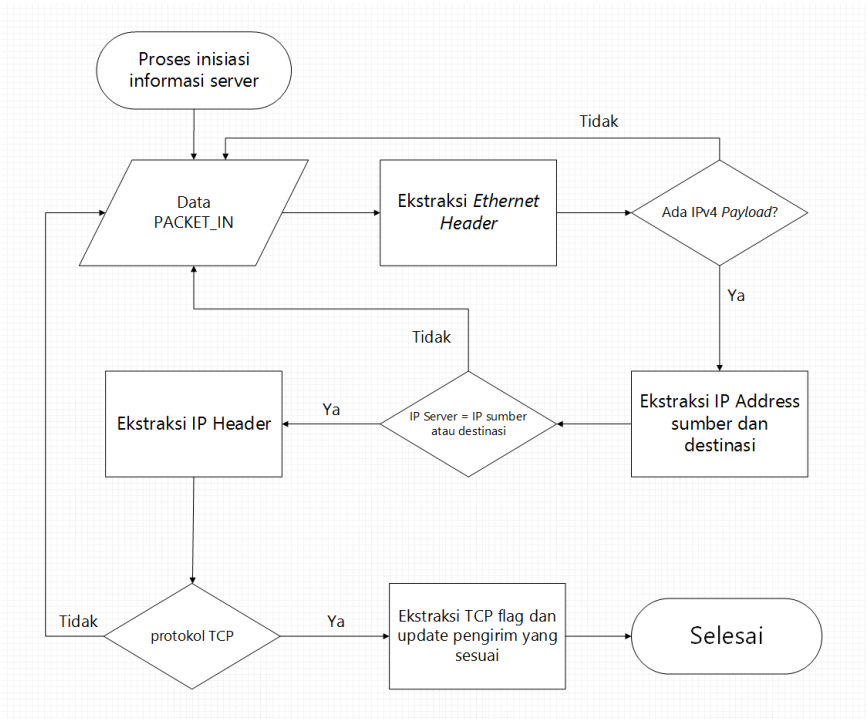
Gambar 1. *Flow* proses DPI

Algoritma yang digunakan dalam implementasi DPI pada arsitektur jaringan SDN ini menggunakan algoritma Aho-Corasick yang menitikberatkan masalah performansi pada *pattern matching* dengan memanfaatkan *tree* yang berisi *set of strings*.

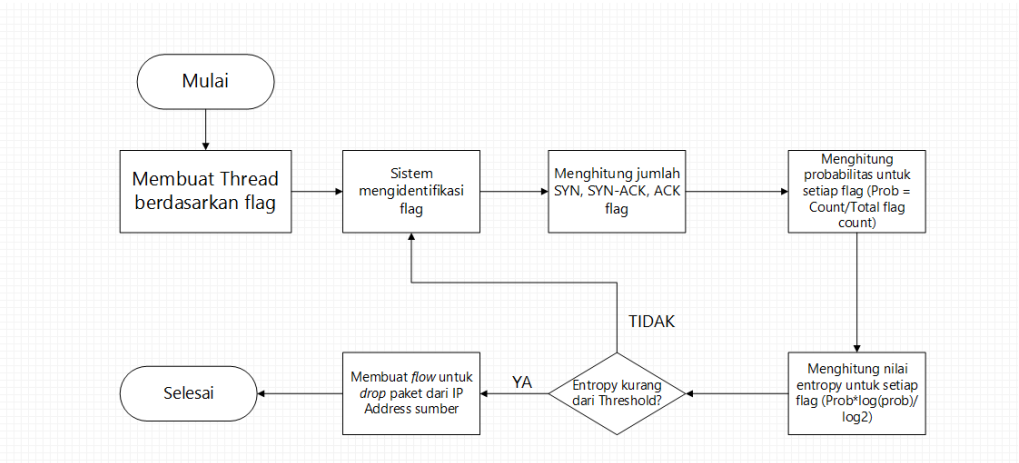
Proses *blocking* dilakukan dengan memanfaatkan program DDoS Detection yang bekerja pada controller Floodlight. Controller menjadi pengendali dari setiap aktifitas yang dilakukan oleh mininet.

Cara kerja dari program DDoS Detection ini sendiri adalah dengan melakukan tiga fase yang terdiri dari:

1) Fase *Collection,* pada fase ini mengoleksi setiap data PACKET_IN yang masuk ke *controller* yang ditempatkan pada fungsi `receive()`. Fase ini di implemntasikan di *class* CDM *(Collection*, *Detection*, dan *Mitigation)*. *Class* CDM mengimplementasikan `IFloodlightModule` dan `IOFMessageListener`. *Flowchart* fase *collection* digambarkan pada Gambar 2.
2) Fase Detection, merupakan fase yang dilakukan pada data yang telah dikoleksi sebelumnya di fase collection. Fase ini juga menentukan bahwa data yang terdeteksi merupakan sebuah ancaman atau bukan dengan membandingkan entropy yang didapat pada informasi perhitungan flag dan threshold yang dibuat sesuai pada sistem [1].
3) Fase *Mitigation,* fase ini akan bekerja saat terjadi serangan. Fase *mitigation* ini menggunakan FlowEntryPusher yang merupakan layanan yang ditawarkan pada *controller* floodlight untuk melakukan *blocking traffic* yang mencurigakan di switch. *Flowchart* dari fase *detection* dan *mitigation* dapat dilihat pada Gambar 3.



Gambar 2. *Flowchart* fase *collection*



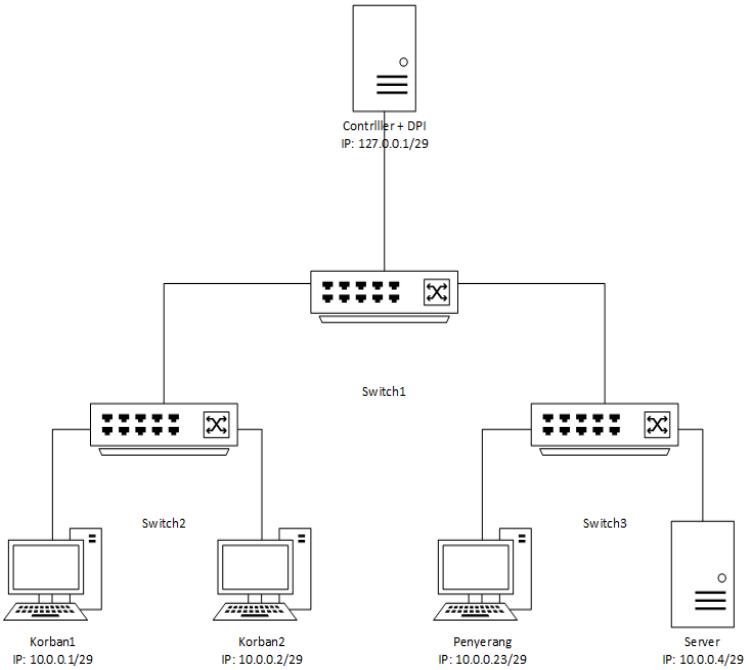
Gambar 3. *Flowchart* fase *detection* dan *mitigation*

C. Rancangan Topologi

Adapun topologi jaringan yang diimplementasikan untuk sistem adalah seperti pada Gambar 4, yang terdiri dari:

1) *Controller* + DPI : adalah sebuah *controller* berupa *tools* Floodlight berfungsi sebagai pusat pengatur jaringan dan DPI untuk melakukan *monitoring*, *analyze*, dan *controlling traffic*
2) Switch : merupakan switch virtual dari mininet sebagai penghubung antar perangkat dalam jaringan yang telah mendukung protokol openflow

3) Server : merupakan server yang memberikan pelayanan DNS Server pada jaringan.
4) Penyerang : merupakan komputer yang digunakan untuk menjalankan scenario penyerangan.
5) Korban : merupakan komputer yang digunakan sebagai target dari serangan.



Gambar 4. Topologi jaringan implementasi

D. Skenario Pengujian

Skenario pengujian yang dilakukan ada dua yaitu saat tidak ada serangan untuk melihat performansi dari *bandwidth* dan *latency* sebelum serangan dan saat ada serangan, dimana dikatakan berhasil saat ntopng yang bertindak sebagai DPI mampu mendeteksi serangan *Denial of Service* (DoS) dengan tipe *Syn Flooding* secara *Direct Attack*. Hasil dari pengujian diolah kedalam grafik pada bagian Analisis Hasil Pengujian dalam Bab empat yang menunjukan paket terdeteksi pada jaringan arsitektur SDN saat sebelum dan sesudah ditambahkan DPI dengan dilihat berdasarkan QoS (*Quality of Service*).

Tujuan : Menguji sistem keamanan terhadap serangan DoS berupa *flood* di jaringan dan melakukan analisa pengaruhnya terhadap *Quality of Service* (QoS).

Deskripsi : *Tools* yang digunakan untuk melakukan *flooding* di jaringan adalah hping, dengan perintah sebagai berikut.

```
hping3 -S –flood server                    (1)
```

Keterangan:
-S, hanya mengirimkan SYN paket saja.
--flood, mengirimkan paket secepat mungkin.
server,ip/hostname tujuan dari penyerang.

Setelah perintah (1) dijalankan di sisi penyerang, maka akan ada notifikasi secara otomatis pada ntopng karena ada indikasi paket berbahaya di jaringan lalu lintas terhadap sistem.

IV. PENGUJIAN DAN ANALISIS

Dalam melakukan analisis dari hasil pengujian ini dibagi menjadi empat subab, yaitu sebelum serangan, sesudah serangan, hasil pengujian, dan analisis hasil pengujian. Hal yang akan dianalisis pada dari pengujian sebelum dan sesudah serangan adalah nilai *bandwidth* dan *latency*, dimana nilai *bandwidth* didapatkan dengan menjalankan perintah (2) di sisi *receiver* pada simulasi, dan menjalankan perintah (3) di sisi Pengirim pada simulasi. Untuk mendapatkan nilai *latency* dari *traffic* digunakan netcat dengan perintah (4).

```
iperf -s                                    (2)
```

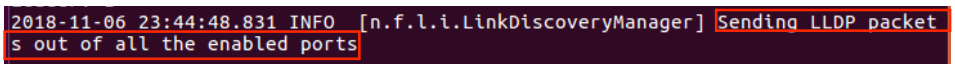
```
iperf -c [IP_Tujuan]                        (3)
```

```
time nc -zw30 [IP_Tujuan]                   (4)
```

Pengujian terdiri dari skenario tanpa adanya serangan dan skenario serangan *Syn Flood* secara *Direct Attack* yang mana dalam pengujianya sebanyak tiga kali uji, untuk Korban1 dan Korban2 dan Server. Dalam setiap pengujian tersebut dilakukan sebanyak sepuluh kali uji untuk mendapatkan hasil yang meyakinkan dengan skenario serangan yang dilakukan.

A. Sebelum Serangan

Dalam subab ini ditampilkan hasil dari pengujian pertama yaitu saat sebelum penyerangan sesuai dengan perencanaan yang sudah dilakukan pada bab tiga bagian Skenario Pengujian. Pada Gambar 5 merupakan informasi yang di dapatkan dari sisi *controller,* yang menjelaskan bahwa semua LLDP (*Link Layer Discovery Protocol*) paket dari semua *port* diaktikan sehingga paket dapat berjalan di jaringan.

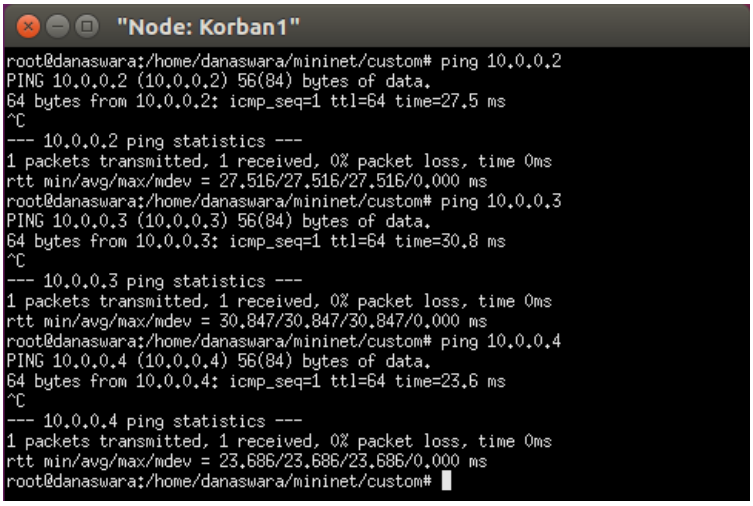


Gambar 5. Status *controller* sebelum serangan

Pada Gambar 6 adalah informasi yang didapatkan pada status *alert* di *tool* ntopng bahwa tidak ada sama sekali *alert* yang terdeteksi oleh *tool.* Hal tersebut menandakan bahwa *host* yang berada di jaringan SDN dapat tetap saling terkoneksi tanpa ada gangguan serangan pada jaringan.

Gambar 6. Infromasi pada ntopng

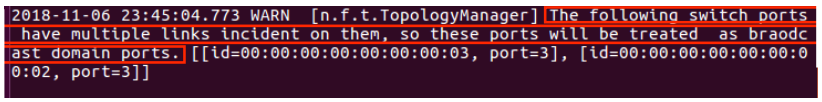
Informasi yang didapatkan dari sisi *host*, pada Gambar 7 ini ketika Korban1 melakukan cek konektivitas kepada Korban2, Penyerang, dan Server adalah koneksi dapat saling tersambung antar *host* pada jaringan yang dibangun. Hal ini berarti menandakan bahwa koneksi dalam jaringan telah tersambung pada masing-masing *host*.



Gambar 7. Konektivitas sebelum ada serangan

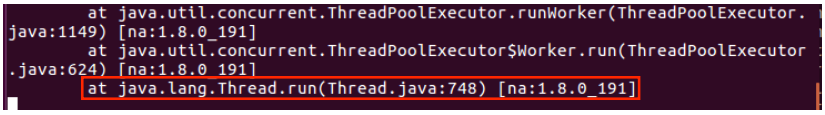
B. Sesudah Serangan

Dalam subab ini menjelaskan mengenai kondisi lalu lintas sesudah dilakukan serangan. Pada Gambar 8 merupakan informasi yang didapatkan pada sisi *Controller* Floodlight sesaat setelah serangan dilakukan, yang menunjukan status bahwa terdapat insiden di switch port dengan multiple links dan *controller* memberlakukannya seperti *broadcast domain ports*. Serangan dilakukan dengan menggunakan perintah (1) pada bab tiga bagian Skenario Pengujian yang dijanlakan di komputer penyerang.



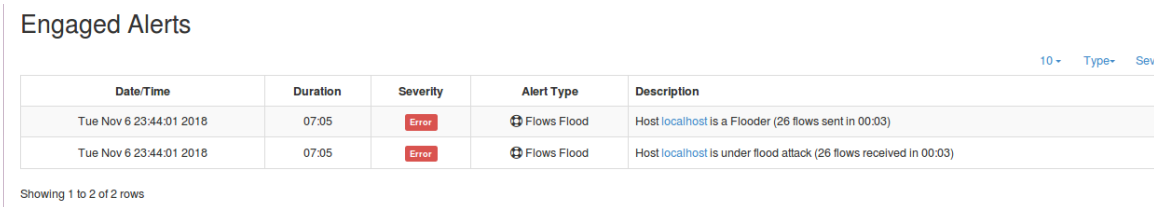
Gambar 8. Status *controller* sesudah serangan

Pada Gambar 9 ada informasi di terminal bahwa *controller* menjalankan thread untuk melakukan reaksi atas apa yang terjadi pada lalu lintas jaringan sesuai hasil yang di deteksi oleh *controller*.



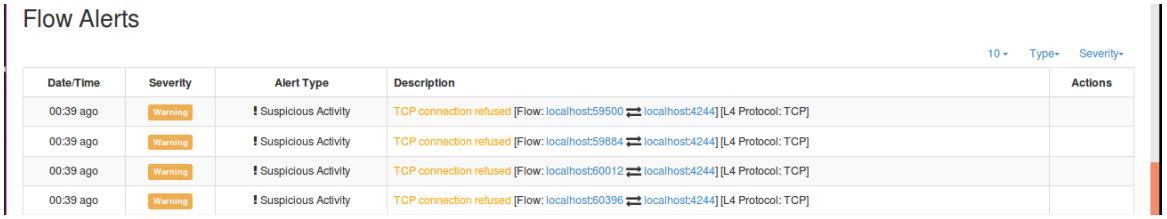
Gambar 9. Status *controller* menjalankan *thread*

Kemudian serangan terdeteksi dan *controller* melakukan reaksi dengan menjalankan fase *mitigation* yaitu pemblokiran. Hal ini juga terdeteksi pada ntopng dengan *warning* seperti pada Gambar 10 yang menunjukan bahwa ada *alerts*. Pada kolom *alert type* ditunjukan bahwa *alert* berupa *flows flood*, kemudian pada kolom *Description* ada keterangan bahwa *alert* yang terjadi ada pada localhost, yang mana localhost disini adalah aktifitas yang dilakukan oleh *host* yang terdeteksi pada mininet.



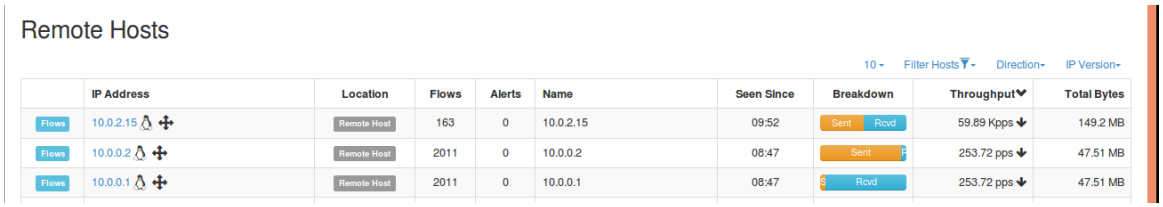
Gambar 10. Status *Alert Flood* di ntopng

Pada Gambar 11 merupakan hasil deteksi dari ntopng mengenai *flow* yang terekam di jaringan secara *real-time*. Pada kolom *alert type* terdapat informasi mengenai *suspicious activity* dan dijelaskan pada kolom *description* bahwa ada koneksi TCP di *refused* dengan informasi *flow* localhost beserta *port*-nya yang berarti *host* pada mininet dan juga *port* yang digunakan yang diperkirakan oleh ntopng merupakan sebuah ancaman.



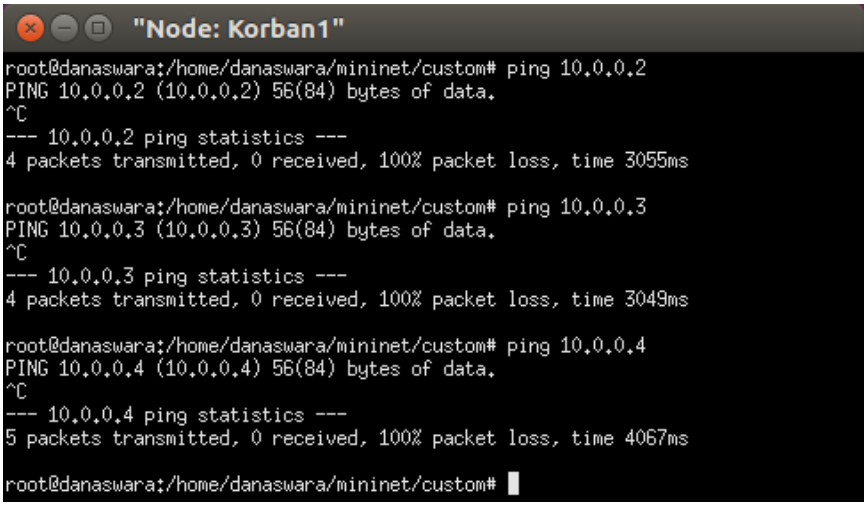
Gambar 11. Deteksi Aktifitas pada *Flow*

Pada ntopng juga dapat dilihat *host* mana yang melakukan banyak aktifitas atau dengan kata lain menggunakan banyak *bandwidth* dalam jaringan (dalam ntopng dikenal dengan *top talkers*) dimana pada kolom IP *Address* merupakan informasi IP *host* dan pada kolom *Total Bytes* adalah informasi mengenai jumlah data yang terekam oleh *host*. Didapatkan urutan tiga besar dari *top talkers* adalah dengan IP Address 10.0.2.15 di urutan pertama yang merupakan IP Address dari *controller,* kemudian IP Address 10.0.0.1 (Korban1) di urutan kedua dan 10.0.0.2 (Penyerang) di urutan ketiga yang merupakan IP Address dari *host* mininet ditunjukan pada Gambar 12.



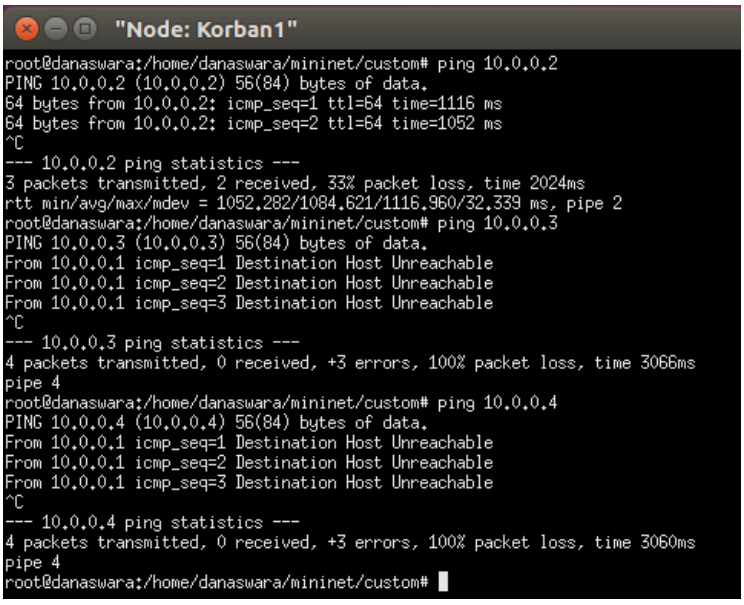
Gambar 12. Top Talkers dari localhost yang menghabiskan *bandwidth*

Saat serangan dilakukan oleh komputer Penyerang ke Korban1 hasil yang didapatkan adalah Korban1 jadi tidak dapat berkomunikasi dengan *host* lainnya yang ada di jaringan (Korban2, Penyerang, dan Server), terbukti dengan informasi 100% *packet loss* yang didapatkan oleh Korban1 ditunjukan pada Gambar 13 yang menandakan bahwa paket tidak tersampaikan dari Korban1 ke masing-masing *host* dalam jaringan.



Gambar 13. Informasi pada Korban1 setelah dilakukan penyerangan ke Korban1

Kemudian, pada saat komputer Penyerang melakukan penyerangan ke Server. Hasil yang didapatkan pada Korban1 dapat dilihat pada Gambar 14. Pada gambar tersebut dapat dilihat bahwa Korban1 masih saling terkoneksi atau dapat tersambung dengan Korban2, sedangkan ke komputer Penyerang dan Server tidak lagi saling terhubung. Hal ini terjadi karena perbedaan *switch,* dimana Korban1 dan Korban2 berada pada Switch2, sedangkan Penyerang dan Server berada pada Switch3. Sehingga saat Penyerang melakukan serangan ke Server, yang mana keduanya berada pada Switch3 sehingga masing-masing *host* yang berada pada Switch2 masih dapat terhubung dalam jaringan.



Gambar 14. Informasi pada Korban1 setelah dilakukan penyerangan ke Server

C. *Hasil Pengujian*

Setelah melakukan dua skenario pengujian, didapatkan data mengenai *bandwidth* dan *latency* yang dilakukan sebelum dan sesudah serangan yang merupakan parameter untuk mengukur QoS (*Quality of*

Service), karena *bandwidth* menjadi gambaran berapa besar data yang dapat melalui jaringan, lalu *latency* yang merupakan waktu yang dibutuhkan hingga data sampai di destinasi.

Pada Tabel 1 merupakan tabel hasil dari pengujian yang dilakukan sesuai dengan rancangan pengujian yaitu sebelum serangan dan setelah serangan ke Komputer Korban1 sesuai dengan perintah (2), (3), dan (4) yang dilakukan pada bab empat mengenai pengujian dan analisis.

TABEL 1
HASIL PENGUJIAN KORBAN1

No	Latency		Bandwidth	
	Sebelum serangan	**Sesudah serangan**	**Sebelum serangan**	**Sesudah serangan**
1	0.017 s	30.03 s	7.64 Mbits/s	0 Mbits/s
2	0.014 s	30.039 s	8.59 Mbits/s	0 Mbits/s
3	0.013 s	30.042 s	8.87 Mbits/s	0 Mbits/s
4	0.01 s	30.041 s	8.64 Mbits/s	0 Mbits/s
5	0.015 s	30.036 s	8.66 Mbits/s	0 Mbits/s
6	0.02 s	30.035 s	8.86 Mbits/s	0 Mbits/s
7	0.009 s	15.749 s	8.58 Mbits/s	0 Mbits/s
8	0.019 s	30.036 s	8.47 Mbits/s	0 Mbits/s
9	0.007 s	30.04 s	8.4 Mbits/s	0 Mbits/s
10	0.018 s	30.033 s	8.68 Mbits/s	0 Mbits/s

Pada serangan yang kedua dilakukan adalah dengan menyerang Korban2 menggunakan skenario SYN *Flooding* yang telah di jelaskan pada subab skenario pengujian. Korban2 di serang sebanyak sepuluh kali dengan mempertimbangkan *latency* dan *bandwidth* sebelum dan sesudah serangan. Hasil dari *latency* dan *bandwidth* pengujian ke Korban2 dapat dilihat pada Tabel 2.

TABEL 2
HASIL PENGUJIAN KORBAN2

No	Latency		Bandwidth	
	Sebelum serangan	**Sesudah serangan**	**Sebelum serangan**	**Sesudah serangan**
1	0.139 s	30.041 s	7.44 Mbits/s	0 Mbits/s
2	0.01 s	30.528 s	7.62 Mbits/s	0 Mbits/s
3	0.02 s	30.167 s	8.11 Mbits/s	0 Mbits/s
4	0.017 s	30.266 s	8.04 Mbits/s	0 Mbits/s
5	0.023 s	30.041 s	7.93 Mbits/s	0 Mbits/s
6	0.009 s	30.036 s	7.60 Mbits/s	0 Mbits/s
7	0.026 s	30.058 s	7.96 Mbits/s	0 Mbits/s
8	0.03 s	30.039 s	7.9 Mbits/s	0 Mbits/s
9	0.01 s	30.123 s	7.45 Mbits/s	0 Mbits/s
10	0.027 s	30.036 s	7.84 Mbits/s	0 Mbits/s

Pada serangan yang ketiga dilakukan adalah dengan menyerang server menggunakan skenario SYN *Flooding* yang telah di jelaskan pada subab skenario pengujian. Server di serang sebanyak sepuluh kali dengan mempertimbangkan *latency* dan *bandwidth* sebelum dan sesudah serangan. Hasil dari *latency* dan *bandwidth* pengujian ke Server dapat dilihat pada Tabel 3.

TABEL 3
HASIL PENGUJIAN SERVER

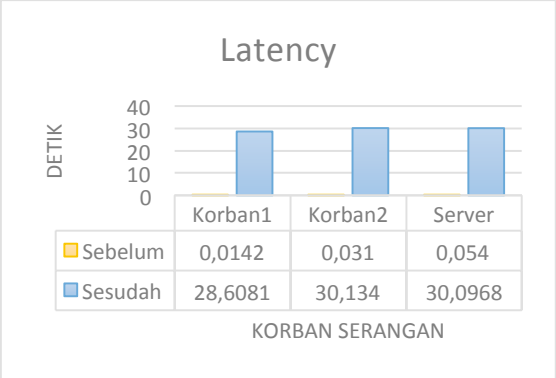
No	Latency		Bandwidth	
	Sebelum serangan	**Sesudah serangan**	**Sebelum serangan**	**Sesudah serangan**
1	0.277 s	30.129 s	6.38 Mbits/s	0 Mbits/s
2	0.023 s	30.068 s	7.39 Mbits/s	0 Mbits/s
3	0.036 s	30.063 s	7.48 Mbits/s	0 Mbits/s
4	0.034 s	30.033 s	8.16 Mbits/s	0 Mbits/s

No	Latency		Bandwidth	
	Sebelum serangan	**Sesudah serangan**	**Sebelum serangan**	**Sesudah serangan**
5	0.035 s	30.216 s	7.61 Mbits/s	0 Mbits/s
6	0.026 s	30.119 s	7.97 Mbits/s	0 Mbits/s
7	0.026 s	30.199 s	7.2 Mbits/s	0 Mbits/s
8	0.032 s	30.041 s	8 Mbits/s	0 Mbits/s
9	0.038 s	30.038 s	7.94 Mbits/s	0 Mbits/s
10	0.016 s	30.062 s	7.89 Mbits/s	0 Mbits/s

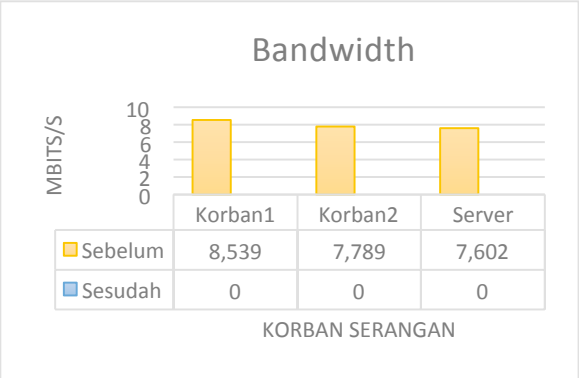
D. Analisis Hasil Pengujian

Parameter QoS (*Quality of Service*) yang digunakan untuk menganalisis hasil pengujian ini adalah *latency* dan *bandwidth* yang telah didapatkan dari pengujian. Kemudian keberhasilan dalam mencegah serangan DoS dengan tipe SYN *Flood* secara *Direct Attack* juga merupakan parameter yang harus di analisis.

Grafik rata-rata dari *latency* dan *bandwidth* sebelum dan sesudah penyerangan dapat dilihat pada Gambar 15 dan Gambar 16. Pada pengujian pertama atau sebelum penyerangan untuk PC Korban1 didapatkan rata-rata dari *latency* sebesar 0.0142 detik, sementara untuk PC Korban2 dan Server masing-masing adalah 0.031 dan 0.054. Kemudian *bandwidth* yang dimiliki oleh PC Korban1, PC Korban2, dan Server masing-masing adalah sebesar 8.539Mbits/s, 7.789Mbits/s, dan 7.602Mbits/s.



Gambar 15. Grafik *latency* di jaringan



Gambar 16. Grafik *bandwidth* di jaringan

Pada pengujian kedua atau setelah serangan, pada PC Korban1, PC Korban2, dan Server terjadi kenaikan *latency* dan penurunan *bandwidth* yang drastis dengan masing masing penaikan rata-rata *latency* sebesar 28.5939 detik, 30.103 detik dan 27.023 detik serta penurunan rata-rata *bandwidth* dari 8.539Mbits/s, 7.789Mbits/s dan 7.602Mbits/s ke 0Mbits/s.

Hal tersebut menunjukan bahwa *latency* meningkat dikarenakan adanya gangguan pada lalu lintas jaringan yang mengakibatkan data dari pengirim membutuhkan waktu yang lebih lama untuk dapat sampai kepada penerima dan juga *bandwidth* yang terjadi setelah serangan adalah sebesar 0Mbits/s yang mengakibatkan data dari pengirim tidak dapat melalui lalu lintas jaringan untuk sampai ke penerima. Hal tersebut menandakan bahwa serangan dapat di deteksi dan diblokir selama penyerangan masih berlangsung di lalu lintas jaringan *Software Defined Networks* (SDN) dengan memanfaatkan *tool ntopng* dan *controller Floodlight* sebagai pendeteksi dan pemblokir.

Kemudian dengan menggunakan ntopng sebagai *tool* DPI, masalah mengenai *false positive* dapat diatasi yang mana ditunjukan pada Gambar 9 dan Gambar 10. Terdapat kolom *severity* pada gambar yang menunjukan tingkat dari *alert* yang diberikan sistem untuk aliran data lalu lintas pada jaringan. ntopng mampu membagi tingkatan ancaman kedalam *warning* dan *error* sehingga masalah *false positive* dapat teratasi.

Penyerangan yang dilakukan sepuluh kali pada jaringan, mendapatkan hasil bahwa setiap paket yang berbahaya pada jaringan akan diblokir oleh sistem sehingga aspek *availability* nya untuk host yang menjadi korban serangan tidak bisa berhubungan dengan *host* lainnya di jaringan, namun jika penyerangan (*host* yang diserang, dan penyerang) berada pada *switch* yang sama, maka *host* lainnya yang berada di *switch* lain dalam jaringan masih dapat terhubung.

V. KESIMPULAN

Dengan berdasarkan pada analisa yang telah dilakukan terhadap skenario serangan DoS dengan jenis SYN *Flood* secara *Direct Attack* kepada tiga *host* yang berjalan di topologi implementasi, dapat diambil beberapa kesimpulan sebagai berikut.

1) *Deep Packet Inspection* (DPI) dapat melakukan *blocking* paket yang dicurigai pada arsitektur jaringan *Software Defined Networks* (SDN) dengan memanfaatkan *ntopng* dan *floodlight* sebagai *controller* yang ditandai dengan meningkatnya *latency* serta *bandwidth* menjadi sebesar 0 Mbits/s pada lalu lintas jaringan.
2) Durasi dari *blocking* paket pada sistem ini dilakukan selama serangan masih terdeteksi di lalu lintas jaringan dengan floodlight sebagai *controller* untuk *collect data*, *detection*, dan *mitigation* dalam melakukan reaksi pada paket yang dicurigai.
3) Masalah *false positive* dapat teratasi dengan menggunakan ntopng sebagai *tool* DPI dimana melakukan pembagian tingkatan serangan yang terdeteksi pada lalu lintas jaringan, sehingga tidak semua data yang dicurigai diblokir oleh sistem.

DAFTAR PUSTAKA

[1] MSJ94. (2016). DDoS Detection in SDN. Github.
[2] Spooner, J. (2016). A Review of Solutions for SDN-Exclusive Security Issues. *(IJACSA) International Journal of Advanced Computer Science and Applications*, *7*(8), 113–122.
[3] Li Yunchun, R. F. (2014). An Parallized Deep Packet Inspection Design in Software Defined Network.pdf. *ICITEC*, 6–10.
[4] ONF. (2015). Principles and Practices for Securing Software - Defined Networks. *White Paper*, (January).
[5] Paul Goransson, C. B. (2014). *Software Defined Networks: A Comprehensive Approach*. Waltham: Morgan Kaufmann.
[6] Zanna, P., O'Neill, B., Radcliffe, P., Hosseini, S., & Ul Hoque, M. S. (2014). Adaptive threat management through the integration of IDS into Software Defined Networks. *2014 International Conference on the Network of the Future, NOF 2014 - Workshop on Smart Cloud Networks and Systems, SCNS 2014*. https://doi.org/10.1109/NOF.2014.7119792
[7] Pratama, R. F., Suwastika, N. A., & Nugroho, M. A. (2018). *Design and Implementation Adaptive Intrusion Prevention System (IPS) for Attack Prevention in Software-Defined Network (SDN) Architecture. 2018 6th International Conference on Information and Communication Technology (ICoICT)*. doi:10.1109/icoict.2018.8528735
[8] Drivers, T., Definition, T., Requirements, K., Broadband, F., & Networks, S. P. (2010). DPI : Deep Packet Inspection Motivations , Technology , and Approaches for Improving Broadband Service Provider ROI, (September).
[9] El-Maghraby, R. T., Elazim, N. M. A., & Bahaa-Eldin, A. M. (2017). *A survey on deep packet inspection. 2017 12th International Conference on Computer Engineering and Systems (ICCES)*. doi:10.1109/icces.2017.8275301
[10] Werlinger, R., Hawkey, K., Muldner, K., Jaferian, P., & Beznosov, K. (2008). The challenges of using an intrusion detection system: is it worth the effort? *SOUPS '08: Proceedings of the 4th Symposium on Usable Privacy and Security*, (1), 107–118. https://doi.org/http://doi.acm.org/10.1145/1408664.1408679
[11] Saad, H. (2016). Deep Packet Inspection using Snort. Victoria: University of Victoria.
[12] Sezer, S., Scott-Hayward, S., Kaur Chouhan P., Fraser, B., Lake, D., Systems Jim Finnegan, C.,… Layout, S. (2013). INTRODUCTION: WHAT IS SOFTWARE-DEFINED NETWORKING? FUTURE CARRIER NETWORKS Are We Ready for SDN? Implementation Challenges for Software-Defined Networks BACKGROUND: WHY SDN? *IEEE Communications Magazine Future Carrier Networks*, *51*(7), 36–43. https://doi.org/10.1109/MCOM.2013.6553676
[13] Kumar, Suresh., Kumar, Tarun., Singh, Ganesh., Maninder, S. N. (2012). Open Flow Switch with Intrusion Detection System. International Journal of Scientific Research Engineering & Technology (IJSRET), 001-004.
[14] Bujlow, T., Carela-Español, V., & Barlet-Ros, P. (2013). Comparison of Deep Packet Inspection (DPI) Tools for Traffic Classification. *Comparison of Deep Packet Inspection (DPI) Tools for Traffic Classification*, 108. Retrieved from http://vbn.aau.dk/ws/files/78068418/report.pdf
[15] Xing, T., Xiong, Z., & Huang, D. (2014). SDNIPS: Enabling Software Defined Networking Based Intrusion Prevention System in Clouds, 2–5.
[16] Zhyuan.Hu, Mingwen.Wang, Xueqiang.Yan, Yueming.Yin, Z. L. (2015). A Comprehensive Security Architecture for SDN. In *International Conference on Intelegence in Next Generation Networks* (pp. 30–37). Paris: IEEE.
[17] Zuma Ibrahim, S. G. (2017). SDN-Based Intrusion Detection System. *INFOTEH-JAHORINA*, *16*.