Tinjauan Keamanan Software-Defined Network berbasis OpenFlow pada Aspek yang berhubungan dengan Kontroler

Tugas Akhir Mata Kuiah Keamanan Informasi dan Jaringan

EL5241

M. Isman Junian R.

23214304

ABSTRAK

SDN merupakan paradigma baru dalam jaringan. Pada jaringan yang ada sekarang, ada dua bagian dalam satu perangkat jaringan yaitu bagian kontrol dan bagian data. Dengan SDN, kedua bagian tersebut terpisah secara fisik sehingga pada perangkat jaringan hanya ada bagian data saja. Untuk dapat mengimbangi jaringan yang ada sekarang, SDN harus dikembangkan paling tidak memiliki fungsi dan fitur yang ada pada jaringan sekarang. Selain fungsi dan fitur, keamanan juga merupakan aspek yang menarik untuk ditinjau dan dikembangkan pada SDN

1. Pendahuluan

Software-defined network (SDN) merupakan salah satu istilah yang baru muncul tahun 2012. SDN merupakan paradigma baru dalam jaringan. Pada jaringan yang ada sekarang, ada dua bagian dalam satu perangkat jaringan yaitu bagian kontrol dan bagian data. Dengan SDN, kedua bagian tersebut terpisah secara fisik sehingga pada perangkat jaringan hanya ada bagian data saja. Paradigma pemisahan bagian kontrol dan data ini sebenarnya sudah ada sejak lama misalnya OPENSIG pada tahun 1995 yang digunakan pada jaringan sirkuit pada PSTN.

SDN dipercaya mampu menggantikan jaringan yang ada sekarang. Jaringan yang ada sekarang bersifat sangat kaku. Jaringan internet yang sekarang telah mengalami ossification sehingga jika tetap menggunakan jaringan sekarang maka pengembangan akan sangat susah. Jaringan berkembang dengan sangat lambat dan tidak signifikan. Berbeda dengan bagian lain pada dunia komputer seperti prosesor dan sistem operasi yang berkembang secara pesat. Perangkat jaringan yang ada sekarang sangat bergantung pada vendor dan banyak sekali protokol *proprietary* sehingga menyulitkan interkoneksi perangkat yang berbeda. Selain itu, manajemen jaringan jauh lebih sulit dibanding manajemen server. Manajemen ratusan server atau ribuan server membutuhkan upaya yang sama karena semua server dapat diatur dengan satu perangkat saja. Sedangkan perangkat jaringan bersifat *closed source* juga terbatasnya API (*application programmable interface*) yang diberikan oleh vendor sehingga butuh upaya yang tinggi untuk mengatur perangkat.

Untuk mengimbangi jaringan yang ada sekarang, SDN harus dikembangkan paling tidak memiliki fungsi dan fitur seperti yang ada pada

jaringan sekarang. Selain fungsi dan fitur, keamanan juga merupakan aspek yang menarik untuk dikembangkan pada SDN.

2. Software Defined Network

Jaringan komputer dibangun dari banyak perangkat seperti switch, dan router yang berfungsi untuk meneruskan paket juga perangkat middlebox semisal *firewall* yang fugsinya untuk manipulasi traffic. Terdapat fenomena "*Internet Ossification*" [1] dimana jaringan internet seolah-olah mengeras(lebih rigid, tidak fleksibel) karena internet merupakan satu infrastruktur yang penting namun dibuat dari sistem yang tertutup sehingga sulit untuk mengembangkannya. Lalu di dalam perangkat-perangkat terdapat protokol kompleks juga setiap perangkat memiliki kontrol masing-masing sehingga operator jaringan memiliki tugas yang sulit semisal mengubah atau menerjemahkan "*policy*" menjadi konfigurasi karena harus mengetahui konfigurasi detail setiap perangkat yang terhubung juga ahli dalam konfigurasi protokol yang kompleks. Untuk memecahkan masalah ini, muncul paradigm baru dalam jaringan yaitu Pemrograman Perangkat Jaringan atau Software-Defined Network yang intinya adalah pemisahan antara bagian kontrol dan bagian *forwarding*.

Software-Defined Network (SDN) menjanjikan manajemen jaringan yang lebih muda dan ringkas juga memungkinkan inovasi juga evolusi. SDN masih tergolong baru dan tumbuh dalam laju yang pesat. SDN memiliki potensi yang sangat besar untuk mengubah cara jaringan beroperasi dan menghapus *middlebox*. Ide untuk memisahkan bagian kontrol dan *forwarding* sudah ada sejak lama misalnya Open Signaling (OPENSIG, 1995, digunakan pada jaringan telepon/*circuit switch*), Active Networking (pertengahan 1990an), Devolved Control of ATM Network (DCAN,

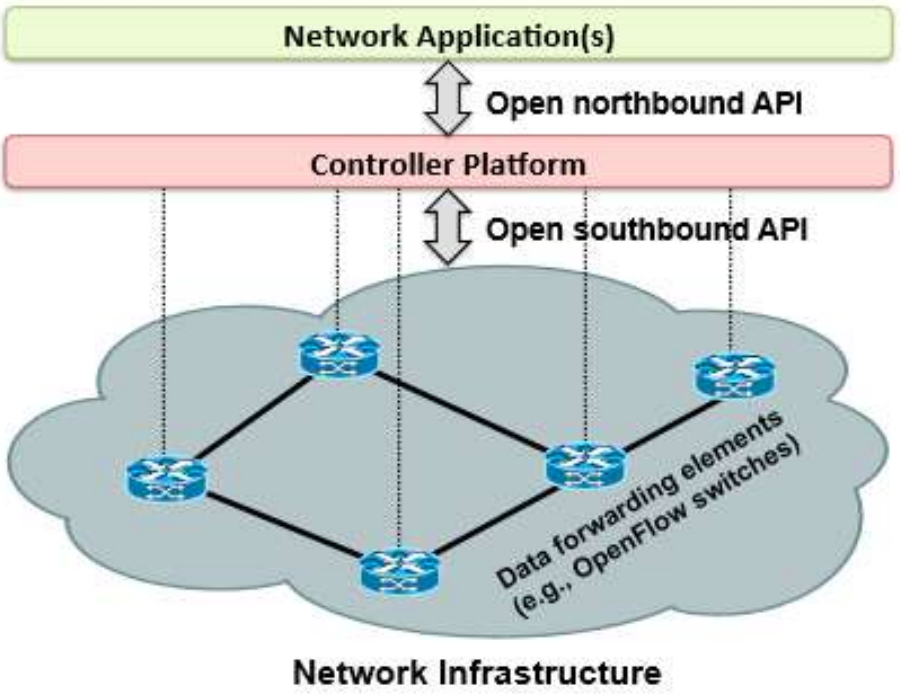
Pertengahan 1990an), 4D Project (2004), NETCONF (2006) dan juga ETHANE (2006).

Setidaknya ada dua arsitektur dalam SDN yaitu ForCES dan OpenFlow. Pada arsitektur ForCES terdapat *Logical Function Block* yang serupa dengan *flow table* pada OpenFlow yang sama-sama berada pada bagian *forwarding* dan diatur oleh bagian kontrol. Namun OpenFlow mendapat dukungan lebih banyak dari kalangan akademisi, periset, dan juga industri. Banyak juga kalangan yang menyebut jika OpenFlow merupakan standar de facto untuk SDN sehingga lebih banyak implementasi dan risetnya.

Pada SDN, *switch* maupun *router* diberi istilah yang sama yaitu perangkat *forwarding* (*forwarding device*). Ada dua jenis perangkat *forwarding* dengan SDN yaitu perangkat murni dan hibrid dimana perangkat hibrid dapat berfungsi seperti perangkat biasa jika tidak terhubung dengan bagian kontrol. Di bagian perangkat *forwarding* ini biasanya terdapat dua prosesor yaitu prosesor untuk tujuan umum (*general purpose processor*, GPP) yang bersifat lambat namun pintar dan khusus (*application specialized integrated circuit*, ASIC) yang bersifat cepat namun tidak pintar. Karena bagian kontrol dipisahkan, GPP pada perangkat *forwarding* dapat dihapuskan namun OpenFlow masih membutuhkan operasi lain disamping *forwarding* seperti memproses *table entries/flow table* atau melakukan *statistical counter*. Operasi tersebut akan susah dilakukan jika dilakukan pada ASIC. Hal ini dapat menjadi tantangan dalam pengembangan perangkat keras SDN.

Pemisahan antara bagian kontrol dan *forwarding* pada SDN dapat diibaratkan dengan sistem operasi dimana bagian kontrol memberi antar muka terhadap jaringan yang dapat digunakan untuk melakukan manajemen dan menawarkan fungsionalitas baru untuk jaringan. SDN dapat diaplikasikan secara luas seperti pada lingkungan jaringan yang heterogen seperti media jaringan ataupun layanannya. Bagian *forwarding* berkomunikasi dengan bagian kontrol melalui *southbound* API. Contoh dari *southbound* API adalah protokol OpenFlow. Jika bagian *forwarding* tidak memiliki *rule* untuk suatu *flow* (data yang masuk) maka bagian *forwarding* akan menanyakan ke bagian kontrol dan kontrol akan memberikan jawaban berupa *rule* untuk diaplikasikan dan dicantumkan dalam *flow table* di bagian *forwarding* bersangkutan.

Pada bagian kontrol pun terdapat tantangan yaitu masalah skalabilitas, model kontrolnya, dan kemanannya. SDN harus memiliki model kontrol yang tingkat *granularity*-nya dapat ditentukan. Lalu bentuk penyebaran kontrolernya dapat berupa *centralized* atau *distributed*, sedangkat sifat *policy*-nya dapat berupa reaktif atau proaktif.

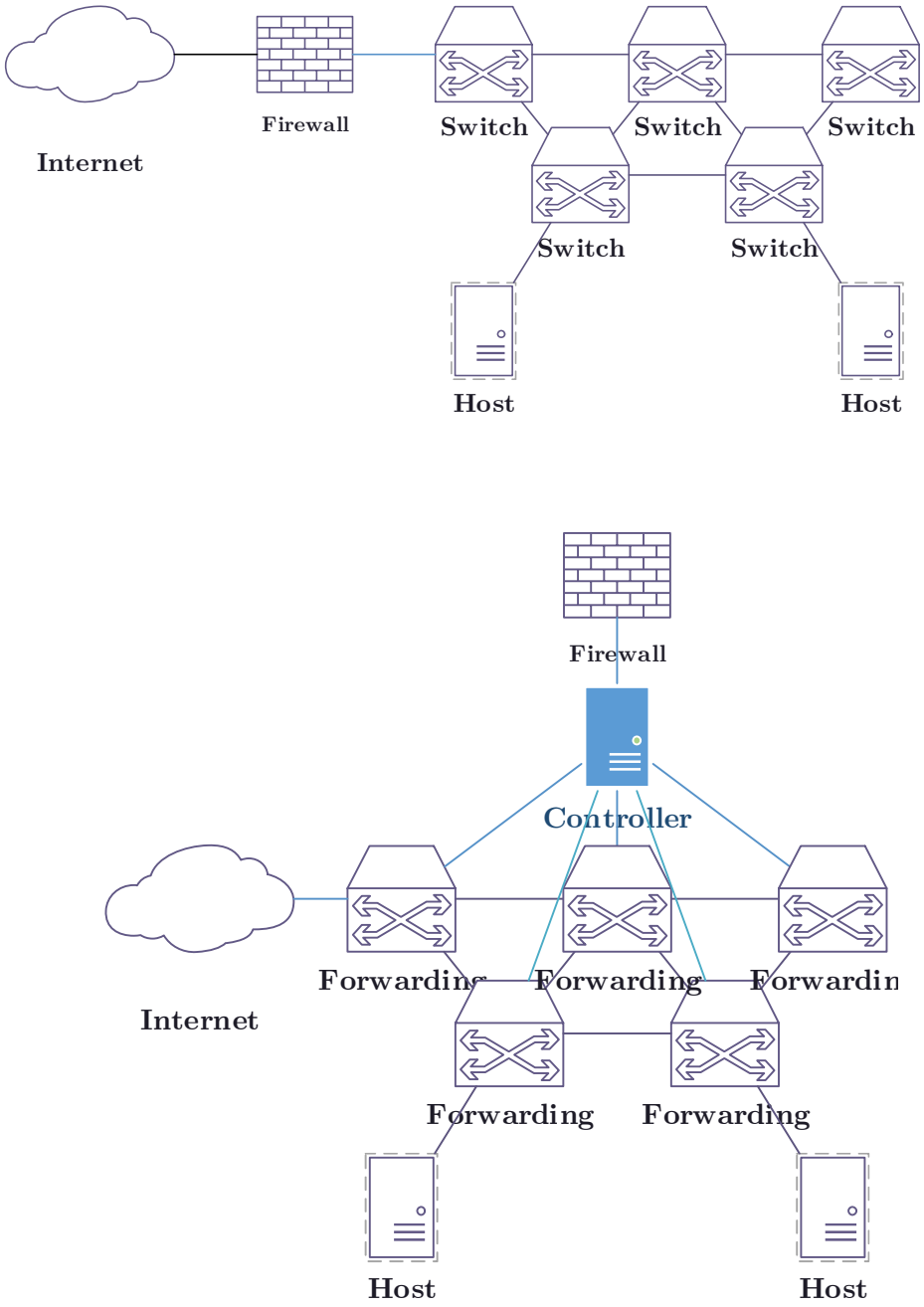


Gambar 1 Arsitektur SDN [2]

Selain dua bagian utama yaitu kontrol dan *forwarding*, ada bagian tambahan yaitu bagian aplikasi. Bagian aplikasi ini akan menggantikan fungsi-fungsi *middlebox* pada jaringan sekarang. *Middlebox* merupakan suatu perangkat yang memiliki fungsi diluar *forwarding* semisal *intrusion detection system*, *firewall*, dan *hardware load balancer*. Beberapa kelemahan dari *middlebox* yang ada sekarang adalah memiliki harga yang mahal, sulitnya menambah fitur karena terkunci vendor, sulitnya manajemen, dan tidak bisa dilakukan *scaling on demand* [3].

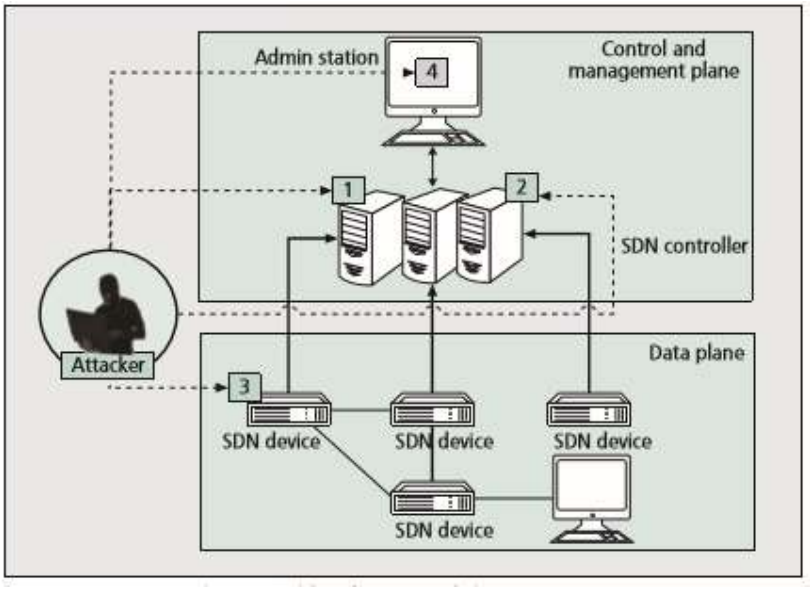
Suatu data center untuk large enterprise dengan pengguna lebih dari 80.000 dan puluhan site, membutuhkan sekitar 636 *middlebox* dan 900 *router*. Total biaya yang dibutuhkan adalah sekitar US$ 10Milyar hanya untuk *middlebox* dibagian keamanan saja. Dengan adanya SDN yang

memiliki *killer application* berupa *virtualization*, biaya bisa dipangkas menjadi sangat rendah.



Gambar 2 Perbedaan topologi pada jaringan konvensional dan SDN. Firewall merupakan aplikasi tambahan yang berjalan diatas kontroler

Ada lima aspek keamanan [4] [5] yang dapat dikaji dalam hal SDN yaitu pada bagian aplikasi, kontroler, *forwarding*, kontroler-aplikasi (northbound) dan kontroler-*forwarding* (southbound). Makalah ini membahas aspek keamanan pada bagian kontroler, kontroler-aplikasi (northbound) dan kontroler-*forwarding* (southbound).



Gambar 3 Beberapa potensi target seragan pada SDN [4]

3. Isu Keamanan pada Kontroler SDN

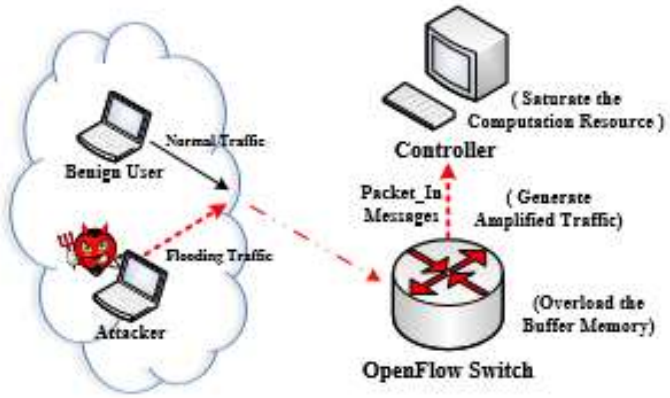
Bagian kontroler dibangun diatas platform *general purpose computing* dan kita semua tahu kelemahan dari platform ini. Bagian ini memiliki kendali utuh untuk jaringan. Bagian kontrol umumnya bersifat tersentralisasi dan masih sedikit penelitian yang membahas kontroler yang bersifat tersebar. Keuntungan dari tersentralisasi adalah berkurangnya area untuk diserang namun jika berhasil diserang maka dampaknya akan sangat besar karena kelemahan dari sentralisasi adalah satu titik kegagalan (*one*

point failure) sehingga jika bagian ini berhasil diretas, maka seluruh jaringan dapat diambil alih.

3.1 Masalah yang Berhubungan dengan Ketersdiaan Sumber Daya

SDN memiliki masalah skalabilitas karena SDN mendorong semua kerumitan (semisal kalkulasi jalur) menuju kontroler dan perangkat *forwarding* hanya melakukan fungsi primitif berupa *forwarding* saja sesuai perintah dari kontroler. Karena semua kerumitan ditangani oleh kontroler maka akan terjadi *bottleneck* pada kontroler. Perangkat *forwarding* maupun kontroler memiliki sumber daya dan pastinya semua sumber daya memiliki batasan. Sumber daya yang ada pada kontroler berupa komputasi, penyimpanan (*buffer*) dan *bandwidth* jaringan.

Salah satu cara yang jitu untuk mengeksploitasi skalabilitas dan menghabiskan sumber daya adalah dengan melakukan serangan denial of service (DoS) sehingga target tidak dapat melayani permintaan dari pengguna. Penyerang hanya perlu membuat paket *malicious* berupa paket dengan header yang bernilai *random* sehingga akan terjadi table-miss pada *forwarding* device karena tujuan tidak ada dalam *flow table*. Ketika terjadi table-miss, *forwarding* device akan meneruskan paket ke kontroler untuk menanyakan langkah selanjutnya. Pada SDN dengan OpenFlow, kejadian ini disebut dengan packet_in. Dengan membuat paket dalam jumlah banyak maka penyimpanan (*buffer*) pada *forwarding* device akan penuh, *bandwidth* pada northbound akan habis oleh anomali packet_in, dan juga sumber daya komputasi pada kontroler akan habis untuk komputasi packet_in yang sangat banyak dalam waktu yang singkat.



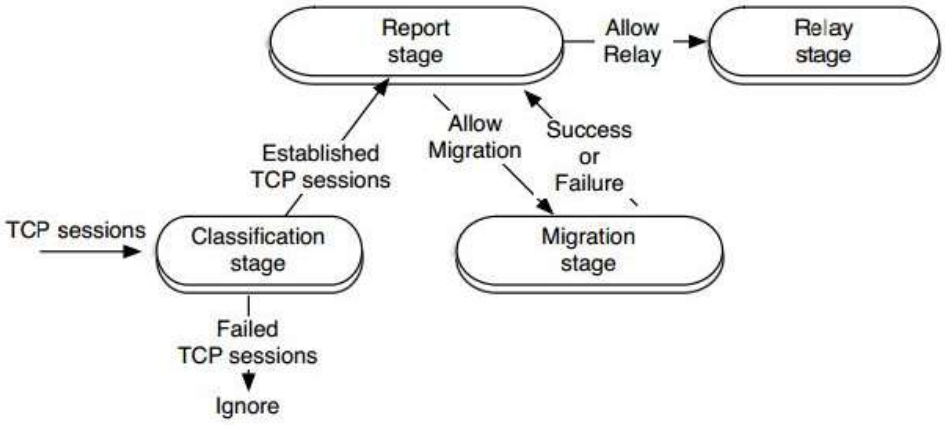
Gambar 4 Ilustrasi DoS 07266854

Karena masalah skalabilitas dan sentralisasi kontrol, maka dengan melakukan serangan melalui beberapa perangakat *forwarding* yang berbeda akan menghabiskan sumber daya pada kontroler dengan sangat cepat.

SDN berbeda dengan jaringan konvensional yang ada sekarang seperti terlihat pada gambar 2. Pada jaringan konvensional, semua paket harus melewati *firewall* secara fisik dan jika telah terhalang oleh firewall maka paket tersebut tidak akan mengganggu jaringan tersebut. Sedangkan pada SDN, *firewall* berada diatas kontroler sehingga jika ada serangan seperti DoS maka akan ada sumber daya akan dikonsumsi oleh serangan DoS tersebut.

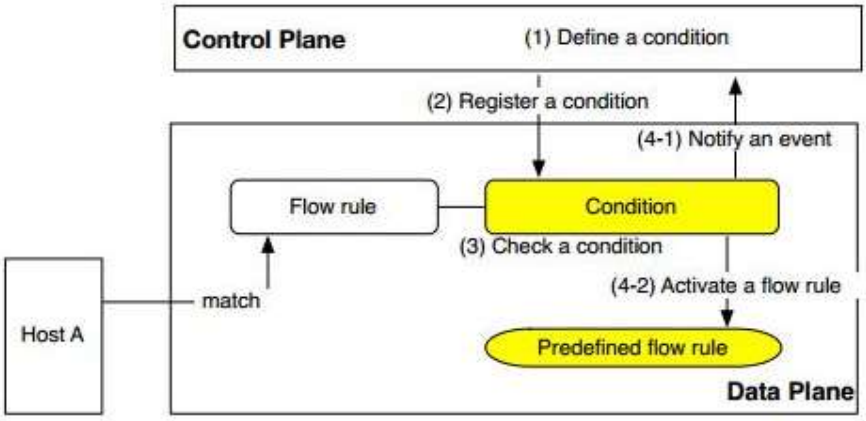
Salah satu dari solusi masalah ini adalah dengan menambahkan kontroler kemudian mengelompokkan kontroler tersebut secara horizontal maupun vertikal. Pengelompokan secara horizontal berarti menempatkan semua kontroler dalam kedudukan yang sama sedangkan pengelompokan secara vertikal berarti pengelompokan berhirarki. Pengelompokan dilakukan berdasarkan fungsionalitas dimana kontroler di bagian bawah akan menangani lebih banyak kejadian yang bersifat lokal sedangkan kontroler di bagian atas akan menangani kejadian global.

Solusi lainnya adalah menambahan sebuah modul kedalam kontroler agar lebih cerdas. Salah satunya adalah dengan AVANT-GUARD [6]. AVANT-GUARD memungkinkan bagian kontrol untuk lebih kebal terhadap serangan DoS.



Gambar 5 Algoritma communication migration pada AVANT-GUARD

AVANT-GUARD terdiri dari dua modul. Yang pertama adalah modul (*communication migration*) migrasi komunikasi yang memungkinkan peningkatan ketahanan jaringan SDN dengan cara menginspeksi sesi TCP pada bagian *forwarding* sebelum melakukan notifikasi pada kontroler. Modul kedua adalah *actuating trigger* yang mememungkinkan kontroler untuk mendeteksi dan merespon ancaman. Modul ini direalisasikan melalui pengumpulan statistik jaringan secara efisien yang secara otomatis akan mengatur *flow rules* sesuai statistik jaringan.



Gambar 6 Algoritma actuating trigger pada AVANT-GUARD

3.2 Masalah yang Berhubungan dengan Hak Akses

Penyerang dapat melakukan elevasi terhadap hak aksesnya semisal mengubah dirinya dari *user* menjadi *admin* sehingga penyerang memiliki akses yang bukan haknya. Penyerang juga dapat melakukan serangan terhadap perangkat yang memiliki akses terhadap kontroler seperti pada gambar 2 dengan vektor serangan nomor 4 pada gambar 3 sehingga penyerang mendapatkan kendali untuk perangkat admin.

Selain itu, akan ada banyak aplikasi yang berjalan dan sangat memungkinkan jika aplikasi-aplikasi tersebut akan memberikan *rule* yang bersinggungan sehingga dibutuhkan keamanan komunikasi pada northbound (antara kontroler dan aplikasi) agar *rule* yang bersinggungan tersebut dapat diatasi. FortNox [7] memberi solusi untuk masalah ini yaitu dengan mengusulkan pembedaan hak berdasarkan peran. Ada 3 peran berbeda untuk *rule insertion requestor* yaitu *human administrator*, aplikasi keamanan, dan aplikasi non-keamanan. *Human administrator* memiliki hak/prioritas paling tinggi, aplikasi keamanan memiliki hak/prioritas sedang sedangkan aplikasi non-keamanan memiliki hak/prioritas paling

rendah. Masing-masing peran diimplementasikan melalui skema tanda tangan digital. Maka ketika terjadi konflik, maka FortNOX akan memilih atau menulak rule sesuai *rule insertion requester*.

3.3 Masalah yang Berhubungan dengan Komunikasi

Mekanisme autentikasi dapat memastikan identitas antar pihak yang berkomunkasi. Pada SDN, bagian kontrol berkomunikasi dengan bagian *forwarding* dan bagian kontrol berkomunikasi juga dengan bagian aplikasi. Dari simulasi SDN yang dilakukan pada mininet versi 2.2.1 dengan POX sebagi kontrolernya, tidak ada mekanisme autentikasi antara perangkat *forwarding* dan kontroler sehingga jika alamat kontroler diketahui maka siapapun dapat menambahkan perangkat *forwarding* baru.

```
mininet@mininet-vm:~$ sudo mn --controller=remote,ip=192.168.192.17,port=6633
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
```

Gambar 7 Perintah untuk menjalankan mininet agar perangkat forwarding dapat dikontrol dari kontroler remote dan tidak membutuhkan autentikasi

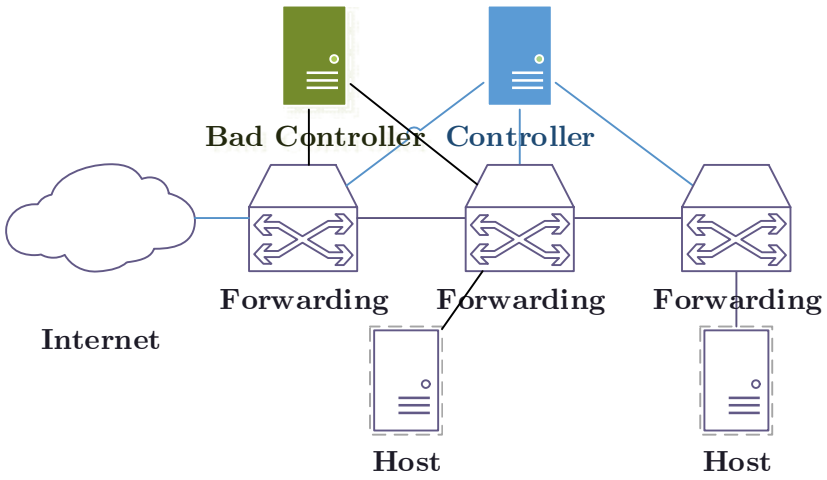
```
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:core:POX 0.2.0 (carp) is up.
INFO:openflow.of_01:[None 1] closed
INFO:openflow.of_01:[00-00-00-00-00-02 2] connected
INFO:openflow.of_01:[00-00-00-00-00-01 3] connected
```

Gambar 8 Pesan verbose pada kontroler yang menyatakan ada forwarding device yang terhubung dengan kontroler. Pada gambar tersebut ada dua perangkat forwarding yang terhubung.

```
mininet@mininet-vm:~$ netstat -atn
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:6634            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:6635            0.0.0.0:*               LISTEN
```

Gambar 9 Perangkat forwarding yang berjalan pada mininet menggunakan port mulai dari 6634. Baik dalam simulasi ataupun kenyataannya, semua perangkat forwarding berjalan pada state LISTEN sehingga membuka celah untuk dieksploitasi

Namun ada ancaman yang lebih serius yaitu pembuatan dan penyusupan kontroler palsu dimana penyerang cukup melakukan *spoofing* IP pada kontroler palsu dengan menggunakan IP kontroler sebenarnya sehingga seluruh jaringan dapat diambil kendalinya. Jika ada penambahan autentikasi, maka perangkat *forwarding* belum tentu dapat terhubung dengan kontroler palsu karena dibutuhkan juga basis data untuk autentikasi.



Gambar 10 Penyusupan kontroler palsu dengan spoofing IP akan mudah karena tidak adanya proses autentikasi antar perangkat

Pada SDN dengan OpenFlow, penggunaan TLS bersifat opsional dan tidak ada definisi untuk standar TLS-nya. TLS sendiri memungkinkan sertifikat antar pihak yang berhubungan dapat dicek secara benar dan sah. TLS dapat mencegah *man-in-the-middle attack* seperti pengubahan isi paket. Selain itu, TLS dapat meningkatkan masalah kerahasiaan data melalui enkripsi. Data yang dipertukarkan antar kontroler dan aplikasi maupun kontroler dan data sangat mungkin mengandung informasi yang sangat sensitif seperti topologi jaringan bahkan data tentang pengguna itu

sendiri. Selain mengubah isi paket, penyerang juga dapat membuat paket palsu yang berisi perintah untuk menambahkan *rule*.

4. Kesimpulan

SDN terus dikembangan oleh peneliti sehingga mampu menyamai bahkan melebihi jaringan yang ada sekarang. SDN tentunya memiliki kelebihan dibanding jaringan konvensional yang ada sekarang seperti mudahnya manajemen perangkat karena kontrol yang tersentralisisi. Tentunya SDN juga memiliki kekurangan diantaranya kelemahan yang muncul karena kontrol tersentralisasi yang menyebabkan *bottleneck* dan menjadi sasaran untuk serangan DoS. Akibat kontrol yang tersentralisasi maka akan membuat sistem menjadi *single point failure* semisal ketika kontroler diretas maka seluruh jaringan dapat dikuasai. Untuk menanggulanginya, dibutuhkan kontroler yang tersebar namun masih sangat sedikit penelitian yang membahas ini.

Pada simulasi SDN yang dilakukan pada mininet, SDN masih kekurangan proses autentikasi untuk setiap perangkat dan juga keamanan komunikasi antar perangkat masih rendah semisal penggunaan TLS yang bersifat opsional dan tidak ada standarnya sehingga informasi yang dikirim dalam bentuk *plaintext* yang dapat mudah dibaca jika jaringan komunikasi tersebut berhasil disadap. Beberapa alasannya adalah kompleksitas untuk implementasi TLS.

Daftar Pustaka

[1] B. N. Astuto, M. Mendonca, X.-N. Nguyen and K. Obraczka, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," *Communications Surveys and Tutorials,* no. 16, pp. 1617-1634, 2014.

[2] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky and S. Uh, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE,* vol. 103, no. 1, pp. 14-76, 2015.

[3] Mulyana, E. , "Virtualization & Software Defined Networking [diambil dari Materi Perkuliahan SDN]," 2015, http://eueung.github.io/EL5244 .

[4] A. Akhunzada, E. Ahmed, A. Gani, M. K. Khan, M. Imran and S. Guizani, "Securing Software Defined Networks: Taxonomy, Requirements, and Open Issues," *IEEE Communications Magazine,* pp. 36-44, April 2015.

[5] S. Scott-Hayward, G. O'Callaghan and S. Sezer, "SDN Security: A Survey," in *IEEE SDN for Future Networks and Services (SDN4FNS)*, Trento, 2013.

[6] S. Seungwon, V. Yegneswaran, P. Porras and G. Gu, "AVANT-GUARD: Scalable and Vigilant Switch Flow Management in Software-Defined Networks," in *ACM SIGSAC conference on Computer & communications security*, 2013.

[7] Porras, P. et al., "A security enforcement kernel for OpenFlow networks," in *ACM SIGCOMM Workshop HotSDN*, Agustus 2012.