Simulasi Aplikasi untuk Mendeteksi dan Mencegah Serangan DDoS pada Jaringan Berbasis Software Defined Network

Sugiyanto Goutama[1], Agustinus Noertjahyana[2], Henry Novianus Palit[3]
Program Studi Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
Jl. Siwalankerto, 121-131 Surabaya 60236, Indonesia
Telp.(031) 2983455 , Fax.(031) 8417658
sugiyantogoutama@gmail.com[1], agust@petra.ac.id[2], hnpalit@petra.ac.id[3]

ABSTRAK

Software Defined Network (SDN) adalah salah satu pengembangan teknologi pada *computer network*. *Computer Network* saat ini umumnya menggunakan banyak *network device*, dimana tiap *network device* memiliki 2 fungsi yaitu *control plane* dan *forwarding plane*. Pemisahan kedua fungsi tersebut pada teknologi SDN, memiliki kelebihan karena desain *control plane* terpusat sehingga mudah melakukan konfigurasi dan manajemen. Namun juga terdapat tantangan berupa *single point of failure* yang rentan pada serangan *Distributed Denial of Service* (DDoS). Oleh karena itu, SDN membutuhkan *Intrusion Detection System* (IDS) dan *Intrusion Prevention System* (IPS) agar mampu mendeteksi dan mencegah serangan DDoS. Penelitian ini bertujuan untuk mengetahui tingkat akurasi dan lama waktu mendeteksi (*Mean Time To Detect*), serta lama waktu mitigasi (*Mean Time To Respond*) dalam menangani variasi serangan DDoS pada topologi SDN. Penelitian ini melakukan pendeteksian dengan dua cara yaitu, *Signature* dan *Anomaly*. Cara *Anomaly* yang akan mengadopsi *Deep Neural Network model* untuk mengklasifikasi, mengenali jenis dan pola serangan DDoS dari sebuah *dataset* dengan beberapa *feature*. Hasil pengujian simulasi dengan 3 jenis serangan yaitu *ICMP Flood*, *SYN Flood* dan *UDP Flood* pada SDN, pendeteksian dengan *signature*-IDS mendapatkan hasil MTTD dan MTTR sebesar 7.2475 detik dan 11.74 detik pada serangan *ICMP*, 26.995 detik dan 11.00 detik pada serangan *SYN*, 20.49 detik dan 3.00 detik pada serangan *UDP*. Sedangkan pendeteksian dengan *anomaly*-IDS tidak digunakan perhitungan berdasarkan MTTD dan MTTR karena cara kerja dari sistem hanya dapat melakukan klasifikasi per satu *packet*. Sehingga dihitung berdasarkan tingkat kesalahan klasifikasi *packet* serangan (*False Negative*) yaitu 7 packet dari total 455 packet pada serangan *ICMP*, 557 *packet* dari total 940 packet pada serangan *SYN,* dan 2 *packet* dari total 3120 packet pada serangan *UDP*. Maka dari untuk *Anomaly*-IDS menggunakan *Deep Neural Network* model masih belum optimal dan masih perlu diteliti dan dikembangkan lebih lanjut.

Kata Kunci: *Software Defined Network* (SDN), IDS, IPS, DDoS, *Mean Time To Detect*, *Mean Time To Respond*

ABSTRACT

Software Defined Network (SDN) is one of the technological developments in computer networks. Today's computer networks generally use many network devices, where each network device has 2 functionalities, called control plane and forwarding plane. The separation of the two functions through SDN technology has the advantage of having a centralized control plane design is to make configuration and management easier. However, there is also a challenge in the form of a single point of failure that is vulnerable to Distributed Denial of Service (DDoS) attacks. Therefore, SDN requires an Intrusion Detection System (IDS) and an Intrusion Prevention System (IPS) to be able to detect and prevent DDoS attacks. This study aims to determine the level of accuracy and length of time to detect (Mean Time To Detect), as well as length of time to mitigate (Mean Time To Respond) in dealing with variations of DDoS attacks on SDN topology. This study detects in two ways, first using a signature and anomaly which will adopt the Deep Neural Network model to classify, recognize the types and patterns of DDoS attacks from a dataset with several features. The results of simulation testing with 3 types of attacks, namely ICMP Flood, SYN Flood and UDP Flood on SDN, detection with signature-IDS get MTTD and MTTR results of 7.2475 seconds and 11.74 seconds for ICMP attacks, 26.995 seconds and 11.00 seconds for SYN attacks, 20.49 seconds and 3.00 seconds on a UDP attack. While the anomaly-IDS detection does not use calculations based on MTTD and MTTR because the workings of the system can only classify per packet. So it is calculated based on the level of misclassification of the attack packet (False Negative), namely 7 packets out of 445 packets for ICMP attacks, 557 packets out of 940 packets for SYN attacks, and 2 packets out of 3120 packets for UDP attacks. Therefore, for Anomaly-IDS using the Deep Neural Network model, is still yet optimal and needs to be researched and developed further.

Key Words: *Software Defined Network* (SDN), IDS, IPS, DDoS, *Mean Time To Detect*, *Mean Time To Respond*

1. PENDAHULUAN

Software Defined Network (SDN) adalah salah satu pengembangan teknologi pada bidang *computer network*, dimana SDN mencoba untuk mengubah arsitektur dari *computer network* saat ini yang dibagi menjadi 2 bagian yaitu *control plane* dan *forwarding plane*. *Control plane* berfungsi untuk menentukan *network packet* yang masuk akan diarahkan ke tujuan, sedangkan *forwarding plane* bertujuan untuk menyimpan sementara dan meneruskan *network packet* dari *control plane* ke suatu tujuan tersebut. Pada SDN akan memisahkan antara *control plane* dan *forwarding plane* dengan tujuan mengatasi permasalahan pada *control plane* yang seringkali mengalami kendala untuk melakukan konfigurasi atau manajemen. Dengan adanya pemisahan pada *control plane* dan *forwarding plane* dalam SDN, maka *control plane* dapat lebih fleksibel dalam manajemen dan konfigurasi karena *control plane* yang bersifat *central* dan berupa *software.*

Salah satu isu / tantangan yang muncul pada SDN adalah dengan terpisahnya *control plane* dan *forwarding plane* pada satu *network device,* maka SDN *controller* yang berperan sebagai *control plane* akan menjadi sebuah *single point of failure*. Hal ini disebabkan oleh beberapa *forwarding plane* yang akan melakukan komunikasi / *request* secara terus menerus sehingga akan mengakibatkan SDN *controller* melambat atau satu jaringan tersebut tidak bisa berjalan dengan baik. Selain itu jika

terdapat DDoS yang masuk ke dalam jaringan, maka jaringan tersebut lebih rentan / berbahaya. Mengingat serangan DDoS merupakan serangan yang paling sering terjadi dalam *computer network*, dimana sejumlah *network packet* akan dikirimkan secara terus-menerus ke sebuah LAN / server dengan tujuan mengurangi ketersediaan *resources / services* dari server atau jaringan yang dituju.

Maka dari itu diperlukan adanya sebuah aplikasi / program *Intrusion Detection System* (IDS) pada SDN, yang bertujuan mendeteksi / mengenali pola serangan untuk mengatasi serangan DDoS. *Intrusion Detection* pada SDN dapat diimplementasikan dalam bentuk *machine learning (ML) model* yang berperan sebagai *Anomaly*-IDS, dimana ML model ini akan melakukan klasifikasi berdasarkan hasil *training* dari sebuah dataset yang diberikan. Namun seperti yang umum diketahui bahwa dalam sebuah dataset pasti memiliki beberapa *feature* / parameter yang digunakan untuk mengelompokkan / klasifikasi beberapa jenis packet (*label* / *target feature*). Secara tidak langsung jumlah data dan jumlah *feature* yang diberikan dari dataset untuk pembelajaran ML model akan mempengaruhi kemampuan model dalam melakukan klasifikasi, serta jenis serangan DDoS yang terus berkembang membuat perlunya pembelajaran dan analisis pola serangan DDoS secara terus-menerus.

Maka dari itu dalam penelitian ini akan melakukan simulasi dengan mengadopsi *Deep Neural Network* (DNN) *model* untuk mengklasifikasi, mengenali jenis dan pola dari serangan DDoS berdasarkan sebuah dataset dengan memakai beberapa *feature*, serta pembuatan aplikasi untuk pendeteksian dan pencegahan serangan DDoS pada jaringan SDN.

2. DASAR TEORI

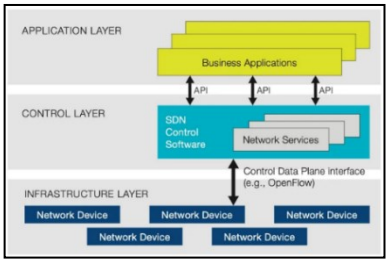
2.1 *Software Defined Network* (SDN)

Software Defined Network (SDN) adalah perkembangan dari jaringan komputer, dimana secara umum sebuah *network device* memiliki dua bagian / *network function* yaitu *control plane* dan *forwarding plane*. Pada SDN dilakukan pemisahan antara kedua *network function* tersebut, sehingga SDN diharapkan dapat melakukan manajemen dalam *overhead communication*, *bandwidth utilization* dan juga memberikan fasilitas pada *network programming* [5].

Perubahan arsitektur pada SDN yaitu terbagi menjadi tiga bagian *application layer, control layer* dan *infrastructure layer,* detail terdapat pada Gambar 1. Selain itu terdapat SDN *control protocol* yang berfungsi untuk menjadi jalur komunikasi untuk dua *layer* pada SDN *northbound protocol* dan *southbound protocol*. *Northbound protocol* menghubungkan *application layer* dengan *contol layer*, sedangkan yang menghubungkan *control layer* dengan *infrastructure layer* disebut *southbound protocol* [5].

Pada *application layer* berisikan beberapa aplikasi / *service* yang dibutuhkan untuk jaringan tersebut. Contoh *Intrusion Detection System (IDS), Intrusion Prevention System (IPS),* yang dapat diakses melalui halaman *dashboard* ataupun menggunakan REST API. Sedangkan pada *control layer* merupakan bagian yang sangat penting pada SDN karena *layer* ini bertugas untuk menyampaikan informasi dan melakukan control dan memproses untuk semua *network packet* yang ada dalam jaringan tersebut, baik dari *application layer* dan *infrastructure layer*. Kemudian pada *infrastructure layer* yang merupakan sekumpulan perangkat seperti *switch / datapath* yang berperan untuk menerima dan meneruskan *network packet* yang masuk dan keluar dari jaringan tersebut [10].

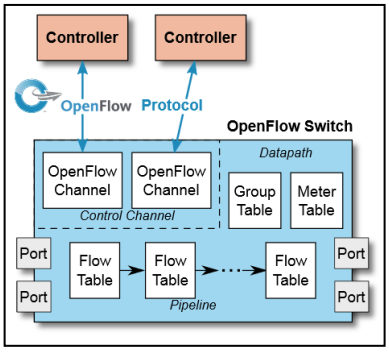
Beberapa kelebihan yang disediakan oleh SDN dibandingkan dengan jaringan tradisional adalah pertama *Agile* yaitu *network admin* dapat menambah ataupun menyesuaikan aplikasi yang dipakai menunjang kebutuhan bisnis atau pengguna, kedua adalah *Centralized management* yaitu SDN menjalankan dan mengkoordinasikan aplikasi yang digunakan secara terpusat, yang memberikan informasi terkait konfigurasi dan aktivitas jaringan yang sedang terjadi sehingga memudahkan *network admin* melakukan *monitoring* atau *troubleshoot*. Lalu ketiga adalah *Network Programming* yaitu kemampuan untuk membuat dan memprogram aplikasi jaringan sesuai kebutuhan [4].



Gambar 1. Arsitektur dari *Software Defined Network*

2.2 *OpenFlow Protocol*

OpenFlow adalah salah satu protokol standar yang digunakan pada jaringan SDN. Protokol OpenFlow ini bertujuan untuk jalur komunikasi antara *control layer* dan *infrastructure layer*. Pada perangkat jaringan yang sudah terpasang protokol OpenFlow akan mempunyai *flow-table* yang berisikan *flow-entity* sebagai *rule* untuk memproses dan meneruskan *packet*, sekaligus mempunyai *openflow-channel* untuk melakukan koneksi dengan SDN controller, detail terdapat pada Gambar 2 Melalui protokol OpenFlow, SDN controller dapat melakukan perintah *add, update* dan *delete* untuk *flow-entry* yang ada dalam *flow-table* tersebut. *Flow-table* berisikan sekumpulan *flow-entry*, dimana sebuah *flow-entry* teridiri dari *match-fields, statistic, priority order* dan *instruction*. Kolom *match-fields* adalah sekumpulan informasi (metadata) yang mendefinisikan sebuah *packet* seperti *source IP-address, destination IP-address, destination Port* dll. Berikutnya kolom *priority order,* kolom ini memiliki *range* nilai maksimal 65.535 dan minimal adalah 0. Kolom ini untuk *indexing* setiap *flow-entity*, yang nantinya akan digunakan dalam proses *matching*. Berikutnya kolom *instruction* adalah bagian yang berfungsi untuk menentukan bagaimana sebuah *network packet* diproses. *Instruction* yang dapat dilakukan untuk satu *flow-entity* adalah salah satu dari 4 pilihan berikut, yaitu mengirimkan *network packet* keluar melalui *port* yang ditentukan, mengabaikan *network packet*, *mengirimkan network packet* ke SDN Controller dan melanjutkan proses *matching* pada *flow-table* berikutnya.



Gambar 2. Arsitektur dari Protokol Openflow

2.3 *Intrusion Detection System* (IDS)

Intrusion Detection System (IDS) adalah sebuah *hardware* atau *software* yang melakukan proses pendeteksian serangan pada sistem atau jaringan. Fungsi utama dari IDS antara lain memantau sebuah sistem / perangkat atau keseluruhan jaringan

terhadap *malicious activity* (contoh : *overloading the network / flooding)* [9]. Pendeteksian serangan dilakukan dengan cara memonitor jumlah / frekuensi *packet* dan menginspeksi *packet header* dan *packet content* pada *network traffic* yang masuk.

IDS memiliki 2 jenis metode untuk melakukan pendeteksian serangan, yaitu *signature-based* dan *anomaly-based*. *Signature-based* juga dikenal sebagai *rule-based*, memiliki cara kerja dengan melakukan inspeksi *packet header (layer 3 dan layer 4)* dan *packet content (layer 7)* untuk setiap *network packet* dengan *signature / rules* yang sudah diketahui. Jika ada kesamaan / kemiripan yang ditemukan dari *network packet* dengan *signature*, maka program / aplikasi IDS akan memberikan *alert / log* kepada *network admin*. *Signature-based* terkadang / sering kali dinilai bersifat kaku (hanya melihat *well-known port*), namun cukup mudah untuk diterapkan. Kumpulan *Signature* perlu diperbarui secara berkala dengan tujuan dapat mendeteksi lebih banyak variasi kemungkinan serangan yang masuk ke dalam jaringan.

Sedangkan untuk *anomaly-based* yang juga dikenal sebagai *behaviour-based*, melakukan deteksi dengan cara mempelajari pola / *behaviour* dari normal *traffic* per satu *packet* selama jangka waktu tertentu dan menghasilkan *profile*. Jika ada abnormal *traffic* (*traffic* yang tidak sesuai dengan *profile*), maka akan memberikan *alert*. Pembelajaran pola tersebut memerlukan jumlah yang cukup banyak dan beberapa parameter pendukung untuk mengklasifikasi / mendeteksi *packet* dengan tepat.

2.4 *Intrusion Prevention System* (IPS)

Intrusion Prevention System (IPS) adalah program keamanan jaringan yang dapat memonitor dan merespon terhadap *intrusion* atau *malicious activity*. Fungsi utama dari IPS adalah melakukan identifikasi aktivitas yang mencurigakan, mencatat informasi kejadian, melakukan upaya untuk memblokir *packet / traffic* dan melaporkan ke *network admin*. IPS memiliki dua macam respon yang umum dipakai untuk menghadapi *intrusion* antara lain pertama adalah *Block Port yaitu* menutup suatu *service port* dalam jangka waktu tertentu, sehingga tidak akan ada koneksi yang terbentuk. Hal ini menyebabkan semua *packet* yang tidak pernah mencapai tujuan / target dalam waktu yang ditentukan. Lalu kedua adalah *Dropping packet* yaitu mengabaikan / membuang paket tanpa memberikan respon pada pengirim atau sumber *network packet*. [8].

2.5 *Distributed Denial Of Services* (DDoS)

Denial of Service adalah salah satu jenis serangan *cyber* yang umum dijumpai. Tujuan umum dari serangan DoS adalah untuk membuat sebuah *service* dan *resource* menjadi habis / tidak bisa beroperasi dengan baik. Cara kerja dari serangan ini yaitu dengan memberikan sejumlah *network traffic* yang sangat banyak. Pada umumnya serangan DoS ini berfokus pada tiga *layer* pada OSI *model*, antara lain *Network Layer, Transport Layer* dan *Application Layer*. Contoh serangan pada *network layer* antara lain *ICMP Floods* dan *Ping Of Death*. Contoh serangan pada *transport layer* antara lain *SYN Flood* dan *UDP Flood* sedangkan pada *application layer* ada *Flood HTTP GET* [12]. Serangan DoS dapat dibagi menjadi 3 kategori yaitu *volumetric attack, protocol attack* dan *application attack* [12]. Pertama *volumetric attack* merupakan serangan DoS yang melakukan *request* dengan jumlah yang sangat banyak, sehingga menghabiskan *bandwidth*. Contoh dari *volumetric attack* adalah *ICMP Flood Attack* dan *UDP Flood Attack.* Lalu untuk *protocol attack* merupakan serangan DoS yang mengeksploitasi protokol TCP yang memenuhi kapasitas koneksi dari target, sehingga tidak dapat memberikan *response* pada *traffic* yang sah.

Melihat dari definisi serangan DoS sebelumnya, maka definisi dari serangan DDoS adalah sejumlah komputer (lebih dari satu) yang melakukan serangan DoS secara bersamaan dan secara terus menerus atau dalam jangka waktu tertentu.

2.6 *Security Operation Center* (SOC)

SOC adalah sebuah regulasi yang dipakai untuk *monitoring* dan *management* yang dapat disesuaikan dengan kebutuhan setiap organisasi. *Key Performance Indicators* (KPI) adalah salah satu matrik penilaian yang dipakai untuk mengukur performa dari SOC, secara umum terdapat dua parameter yang dipakai yaitu nilai *Mean Time to Detect (MTTD)* adalah waktu yang dibutuhkan untuk melakukan validasi serangan yang terjadi pada jaringan, lalu nilai *Mean Time to Respond (MTTR)* yaitu waktu yang dibutuhkan untuk menghentikan serangan atau insiden [7].

Untuk perhitungan MTTD adalah total perbedaan waktu deteksi pertama (munculnya *alert* pertama) dari waktu serangan dimulai, dibagi dengan jumlah total insiden. Sedangkan untuk perhitungan MTTR adalah total perbedaan waktu deteksi terakhir (waktu *alert* terakhir) dari waktu *alert* pertama, dibagi dengan jumlah total insiden. Rumus perhitungan MTTD dapat dilihat pada Persamaan 1 dan MTTR pada Persamaan 2.

$$Mean\ Time\ To\ Detect\ = \frac{\Sigma\ Incident\ detection\ time}{\Sigma\ Number\ of\ Incidents} \quad (1)$$

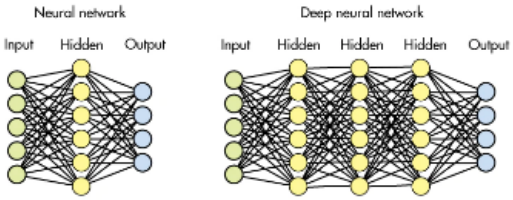
$$Incident\ detection\ time = First\ detection\ time - First\ attempt\ of\ attack$$

$$Mean\ Time\ To\ Respond\ = \frac{\Sigma\ Incident\ response\ time}{\Sigma\ Number\ of\ Incidents} \quad (2)$$

$$Incident\ response\ time = Last\ detection\ time - First\ detection\ time$$

2.7 *Deep Neural Network* (DNN)

Artificial Neural Network (ANN) adalah salah satu contoh *machine learning model* yang melakukan suatu komputasi untuk menghasilkan sebuah *output* atau informasi berdasarkan input / data yang diberikan. *Neural Network* merupakan bentuk atau konsep terapan dari jaringan saraf manusia, dimana setiap bagian (*neuron*) bertanggung jawab untuk mengerjakan komputasi yang kecil. *Neuron* tersebut akan menyalurkan hasil komputasi / data yang diperoleh ke *neuron* yang lain sehingga data tersebut akan dijadikan satu untuk menjadi *output / informasi* (nilai prediksi atau hasil klasifikasi) [13].



Gambar 3. Perbedaan arsitektur ANN model dan DNN model

D*eep Neural Network* (DNN) adalah perkembangan dari ANN yang memiliki jumlah *hidden layer* lebih banyak (lebih dari 2 *hidden layer*) (Gambar 3). Cara kerja dari ANN maupun DNN adalah bersifat *FeedForawrd*, yang berarti akan berjalan satu arah dari *input layer* menuju *output layer* melewati *hidden layer*. Dalam upaya untuk medapatkan sebuah model yang optimal, perlu dilakukan penyesuai bobot / nilai / *weight* dari setiap *node* di *hidden layer*. Upaya tersebut dikenal dengan *back-propagation* yang juga berkaitan dengan *optimizer*. *Optimizer* adalah sebuah teknik *training* dari *back-propagation,* secara umum menggunakan *Adaptive Moment Estimation (ADAM)* atau *Stochastic Gradient Descent (SGD).* Dalam DNN model terdapat 3 jenis *activation function* yang pertama adalah *Tanh Relu Sigmoid* [1]. Berdasarkan sumber [2] menyebutkan bahwa metrik evaluasi yang umum digunakan adalah *threshold metric* yang terdiri dari *accuracy*, *precision* dan *recall*. *Accuracy* adalah suatu

perhitungan untuk dengan menjumlahkan nilai TN dan TP dibagi dengan keseluruhan elemen dari *confusion matrix* (TP, TN, FN, FP) dapat dilihat pada Persamaan 3. *Precison* bertujuan untuk menghitung kemampuan model dalam melakukan klasifikasi terhadap satu data yang relevan dapat dilihat pada Persamaan 4 dan *recall* adalah menghitung kemampuan model dalam melakukan terhadap semua data yang relevan dapat dilihat pada Persamaan 5. Sedangkan Metrik *F1-Score* akan memberikan infomasi sebuah model dengan kemampuan *precision* dan *recall* yang paling baik / mendekati nilai 100 persen dapat dilihat pada Persamaan 6.

$$Accuracy = \frac{TN + TP}{TN + FP + FN + TP} \quad (3)$$

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

$$Recall = \frac{TP}{TP + FN} \quad (5)$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \quad (6)$$

2.8 NSL-KDD Dataset

NSL-KDD adalah *dataset* yang dibuat berdasarkan *dataset* KDD 1999, *dataset* ini dibuat dengan tujuan mengatasi beberapa masalah yang ada pada *dataset* KDD 1999. Jumlah data *training* dan data *testing* pada NSL-KDD adalah 125.973 baris *training* dan 22.544 baris *testing*, lalu memiliki jumlah *feature* yang sama dengan *dataset* KDD 1999 yaitu sejumlah 41 ditambah satu *target feature.*

Feature tersebut dikelompokan menjadi empat kategori yaitu *Intrinsic* didapatkan dengan cara mengambil meta-data / header dari network packet tanpa melihat detail bagian *payload*. *Content* menyimpan informasi tentang paket asli, karena dikirim dalam beberapa bagian, bukan satu. Dengan informasi ini, sistem dapat mengakses *payload*. *Time-based* menyimpan analisis input packet selama dua detik dan berisi informasi seperti, berapa banyak / jumlah koneksi yang dicoba dibuat ke *host* yang sama. Fitur-fitur ini sebagian besar adalah menghitung *statistic* daripada informasi *payload*. Lalu untuk *Host-based* mirip dengan fitur *Time-based*, kecuali fitur ini menganalisis serangkaian koneksi yang dibuat (berapa banyak permintaan yang dibuat ke *host* yang sama melalui x-jumlah koneksi) [11].

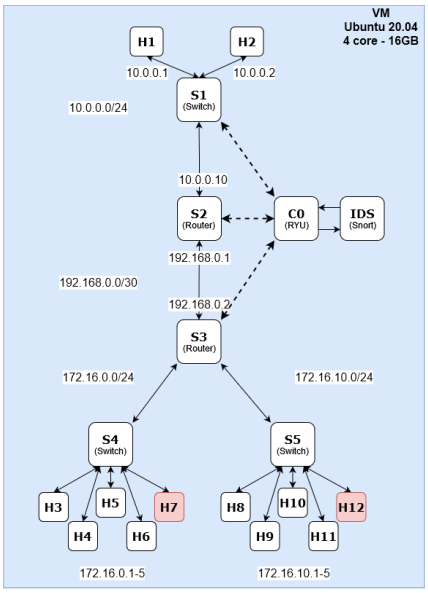
3. DESAIN SISTEM

3.1 Desain Sistem

Desain sistem yang akan dibuat dalam penelitian ini adalah sebuah program simulasi aplikasi untuk mendeteksi dan mencegah serangan DDoS pada jaringan SDN. Dimana akan ada 2 teknik untuk melakukan deteksi yaitu menggunakan *Signature*-IDS dan *Anomaly*-IDS. Pada penelitian ini, teknik yang digunakan untuk mencegah serangan DDoS adalah *drop-packet* dan *block-port*.

Service yang digunakan yaitu H1 akan beperan sebagai UDP server dengan *service* TFTP dan iperf-udp. Kemudian pada H2 akan berperan sebagai TCP server dengan *service* FTP, HTTP, dan iperf-tcp. Jumlah *client host* yang digunakan adalah 10, dari 10 *client host* akan terdapat 8 *host* untuk *traffic* normal mengakses *service* dan 2 *host* untuk melakukan serangan (*host* H7 dan H12). Bentuk topologi yang akan digunakan dapat dilihat pada Gambar 4.

Untuk pendefinisian serangan DoS yang digunakan adalah bersifat *volumetric* sebagai berikut : "X paket dalam jangka waktu Y detik". Pada penelitian ini akan mendeteksi tiga jenis serangan DoS, yaitu : *ICMP Flood Attack*, *SYN Flood Attack* dan *UDP Flood Attack*. Masing – masing dari jenis serangan DoS akan memiliki dua tingkatan yang dapat dilihat pada Tabel 1. Definisi serangan DoS tersebut disesuaikan berdasarkan jurnal [6].

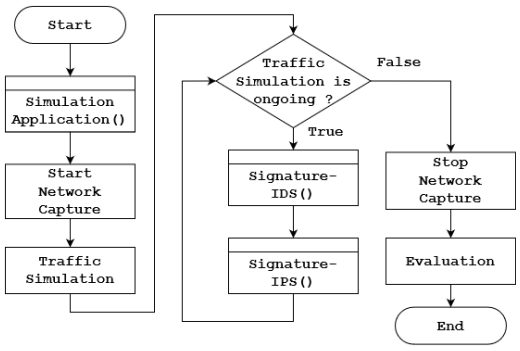


Gambar 4. Topologi SDN

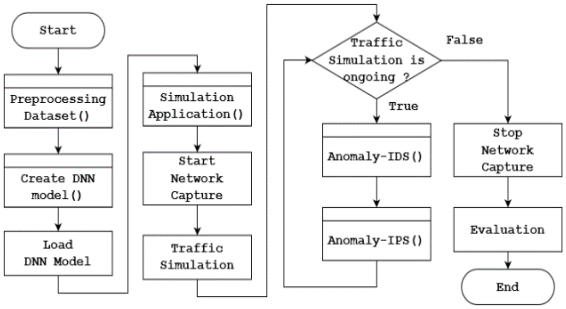
Tabel 1. Definisi DoS secara *signature*-based

Jenis	Level 1	Level 2
ICMP Flood	45.000 Paket dalam 30 detik	90.000 Paket dalam 30 detik
SYN Flood	75.000 Paket dalam 30 detik	150.000 Paket dalam 30 detik
UDP Flood	45.000 Paket dalam 30 detik	90.000 Paket dalam 30 detik

Alur program keseluruhan yang digunakan untuk *Signature*-IDS dapat dilihat pada Gambar 5 dan *Anomaly*-IDS pada Gambar 6.



Gambar 5. Desain Sistem *Signature*-IDS



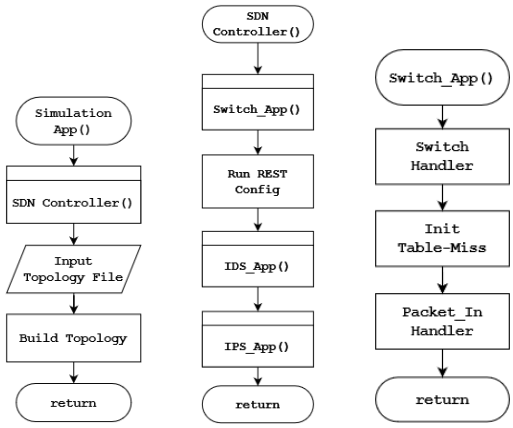
Gambar 6. Desain Sistem *Anomaly*-IDS

3.2 Pembuatan Model DNN

Dataset akan dilakukan *preprosessing* terlebih dahulu dengan melakukan *Label-Encoder* dan *OneHot-Encoder*. Lalu dilanjutkan dengan melakukan *encoding* pada *target feature*, dan memfokuskan pada *target feature* normal dan DoS. Setelah itu melakukan standarisasi menggunakan *Standard-Scaler*, yang bertujuan mengubah *range value* / nilai dari setiap *feature* memiliki standar deviasi yang kecil sehingga dapat mengurangi *overfitting* dan mempercepat proses *training*. Penelitian ini akan membuat 2 model DNN, yang pertama berdasarkan jurnal [3] dan melakukan *feature selection* dengan metode RFE sejumlah 10 persentil. Model DNN yang pertama memakai 8 *feature* (*src_bytes, dst_bytes, hot, num_failed_logins, count, dst_host_count, dst_host_srv_count, dst_host_srv_rerror_rate*) dan hasil *feature selection* adalah 13 *feature* (*src_bytes, dst_bytes, wrong_fragment, num_compromised, same_srv_rate, diff_srv_rate, dst_host_count, dst_host_same_srv_rate, dst_host_serror_rate, dst_host_srv_serror_rate, service_ecr_i, flag_RSTR, flag_S0*). Masing - masing model DNN akan di-training dengan *epoch* 500 dan *monitoring-loss* 100 iterasi, *optimizer* menggunakan ADAM, jumlah *hidden layer* adalah 20 dengan jenis *dense layer*, jumlah *node* setiap *hidden layer* adalah sejumlah *feature* yang digunakan.

3.3 Modul Aplikasi Simulasi

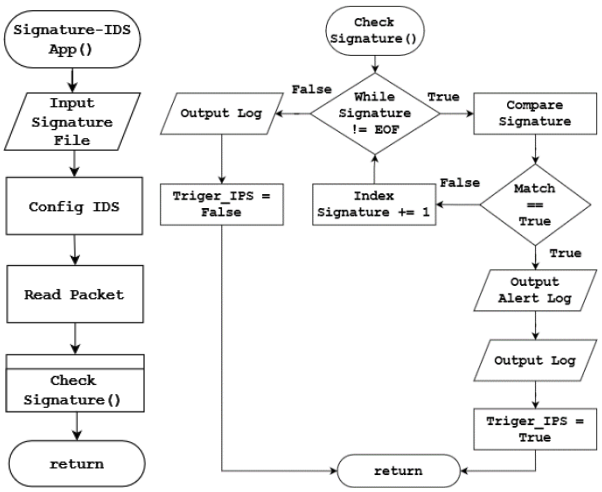
Pada modul ini akan menjalankan fungsi SDN *Controller* (Gambar 7), dimana di dalam fungsi SDN *Controller* terdapat fungsi Switch yang memiliki tujuan untuk untuk melakukan pendaftaran perangkat *switch* yang terkoneksi ke SDN *Controller,* melakukan installasi *tabel-miss* pada masing – masing *switch* yang digunakan, memproses *packet* dan mempersiapkan penambahan *flow-entity* pada *switch* tersebut jika dibutuhkan. Lalu proses *REST config* melakukan konfigurasi *Router* yang akan digunakan pada simulasi (seperti *default-gateway, static-router* dan *default-route*).



Gambar 7. Modul Aplikasi Simulasi

3.4 Modul *Signature*-IDS

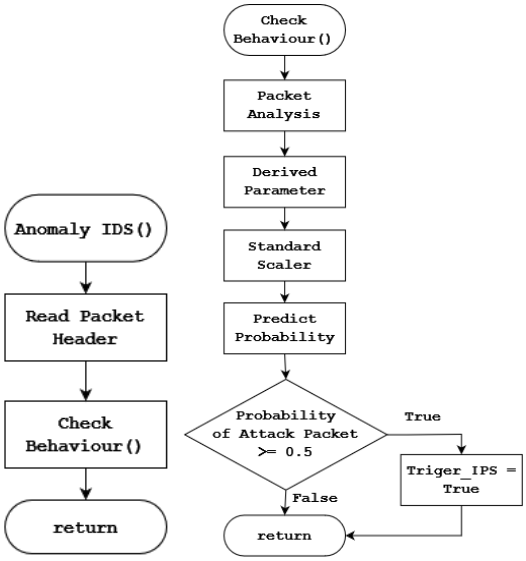
Pada modul ini dimulai dengan memberikan *input signature IDS file* yang akan digunakan untuk simulasi. Setelah itu akan menjalankan IDS, membaca *network packet* yang masuk dilanjutkan dengan melakukan komparasi terhadap semua *signature* yang sebelumnya diberikan dengan cara memanggil fungsi *CheckSignature.* Proses komparasi akan selalu dimulai dari daftar *signature* pertama hingga akhir, jika tidak terdapat *signature* yang *match* hingga *signature terakhir* maka diasumsikan / dinyatakan *packet* tersebut *normal*. Sebaliknya jika terdapat *signature* yang *match*, maka *packet* adalah serangan dan akan memberikan / men-*generate alert* kepada SDN *Controller* juga memberikan *signal* untuk modul IPS. Alur modul *signature*-IDS dapat dilihat pada Gambar 8.



Gambar 8. Modul *Signature*-IDS

3.5 Modul *Anomaly*-IDS

Pada modul ini akan membaca *packet header* yang masuk, lalu melakukan fungsi CheckBehaviour yang memiliki fungsi untuk melakukan analisa *packet*, *derived parameter,* standarisasi menggunakan *Standard-Scaler*, lalu akan melakukan klasifkasi *packet* melalui model DNN yang sudah terbuat. Pada proses analisa *packet* dan *derived parameter* dilakukan menggunakan tool *kdd99-feature-extractor*. Untuk alur modul *anomaly*-IDS dapat dilihat pada Gambar 9.



Gambar 9. Modul *Anomaly*-IDS

3.6 Modul IPS

Modul ini akan dibagi menjadi 2 bagian, yaitu modul *Signature*-IPS dan *Anomaly*-IPS. Modul *Signature*-IPS akan melakukan mitigasi *drop-packet* jika terdapat *alert* pada level 1, sedangkan untuk mitigasi *block-port* terjadi ketika *alert* pada level 2. Sedangkan untuk modul *Anomaly*-IPS, terdapat perbedaan penentuan cara mitigasi yaitu mitigasi *block-port* jika nilai probabilitas serangan dari hasil klasifikasi model DNN label '1' mencapai 0.75. Sedangkan untuk mitigasi *drop-packet* terjadi ketika nilai probabilitas lebih besar dari 0.5.

3.7 Modul Evaluasi

Pada modul ini berjalan setelah tidak ditemukannya *packet* dari simulasi *traffic* yang dirancang, yang kemudian diberhentiankannya proses *network capture*. Kemudian aplikasi

ini akan menerima tiga *input log file* dari *simulation-log*, *mitigation-log* dan *alert-signature.log* ataupun *alert-anomaly.log*. Ketika *log file* tersebut sudah siap, maka akan melakukan perhitungan MTTD dan MTTR berdasarkan ketiga *log file* tersebut. Perhitungan *Mean Time To Detect* (MTTD) (Persamaan 1) yang akan digunakan dalam evaluasi ini untuk mengetahui rata – rata dari lama waktu yang dibutuhkan oleh modul IDS dalam mendeteksi sebuah serangan DDoS. Selain itu juga akan ada pengukuran *Mean Time To Respond* (MTTR) (Persamaan 2) untuk mengetahui rata – rata dari lama waktu yang dibutuhkan oleh modul IPS untuk melakukan *response* / mitigasi terhadap hasil deteksi serangan DDoS dari modul IDS. Selain itu juga akan melakukan perhitungan performa dari model DNN yang digunakan dalam mengklasifikasikan *packet* normal dan *packet* serangan sesuai Persamaan 3 hingga Persamaan 6.

4. PENGUJIAN SISTEM

Pengujian yang dilakukan dalam penelitian ini diawali dengan analisa *feature* yang digunakan oleh model DNN (yang berperan sebagai *Anomaly*-IDS). Selanjutnya akan melakukan perhitungan nilai MTTD dan MTTR menggunakan *Signature*-IDS dan perhitungan nilai tingkat kesalahan dalam mengklasifikasikan *packet*.

4.1 Hasil Analisa *Feature Importance*

Pengujian hasil analisa *feature importance* terhadap *feature* yang digunakan. Melihat dari 8 *feature* yang dipilih dan 13 *feature* hasil *feature selection,* ditemukan irisan *feature*. Irisan *feature* yang didapatkan berjumlah 3 yaitu '*src_bytes', 'dst_bytes'* dan '*dst_host_count'*. Adanya irisan yang ini memperlihatkan bahwa terdapat perbedaan tingkat relevansi dari masing – masing *feature* dalam menentukan klasifikasi. Maka dari itu dilakukan pemeriksaan untuk setiap *feature* yang terpilih menggunakan cara perhitungan *feature importance*.

Tabel 2 adalah hasil *feature importance* dari 8 *feature* yang dipilih, menunjukan bahwa *feature* '*src-bytes'* dan '*count'* ini memiliki tingkat relevansi yang tinggi dalam menentukan klasifikasi *packet* normal dan *packet* serangan yaitu sebesar 0.94. Kemudian melihat nilai dari *feature* '*hot'* dan '*num_failed_logins'* mendapatkan nilai yang cukup rendah, yaitu 0.01152 untuk 'hot' dan 0.0 untuk '*num_failed_logins*'. Sehingga dinyatakan bahwa *feature 'hot'* dan '*num_failed_logins'* ini dapat diabaikan.

Tabel 2. Hasil *Feature Importance* 8 *feature*

0	Score: 0.76248	src_bytes
1	Score: 0.02592	dst_bytes
2	Score: 0.01152	hot
3	Score: 0.00000	num_failed_logins
4	Score: 0.18062	count
5	Score: 0.00476	dst_host_count
6	Score: 0.01290	dst_host_srv_count
7	Score: 0.00180	dst_host_srv_rerror_rate

Tabel 3. Hasil *Feature Importance* 13 *feature*

Ranking	Nama feature	Ranking	Nama feature
1	*dst_bytes*	13	*diff_srv_rate*
1	*dst_host_srv_serror_rate*	15	*num_compromised*
1	*same_srv_rate*	16	*service_ecr_i*
1	*wrong_fragment*	18	*flag_RSTR*
2	*dst_host_serror_rate*	19	*flag_S0*
11	*dst_host_count*	108	*src_bytes*
12	*dst_host_diff_srv_rate*		

Kemudian untuk hasil *feature importance* dari 13 *feature* pada Tabel 3, terdapat 4 *feature* yang memiliki relevansi yang tinggi dalam menentukan klasifikasi *packet* normal dan *packet* serangan yaitu '*dst_bytes', 'dst_host_srv_serror_rate', 'same_srv_rate'* dan '*wrong_fragment'*.

4.2 Hasil Percobaan Model DNN Menggunakan Dataset NSL-KDD

Pada bagian ini menampilkan hasil analisa *testing* dari kedua model DNN yang dibuat berdasarkan *dataset* NSL-KDD. Dari kedua model DNN yaitu dengan 8 *feature* dan 13 *feature*, masing - masing dilakukan percobaan dengan melakukan perbedaan *activation function* (Tanh, ReLu dan Sigmoid). Hasil percobaan yang didapatkan adalah kedua model DNN dengan *activation function* ReLu mendapatkan nilai *accuracy* dan *F1-Score* yang lebih tinggi dibandingkan dengan *activation function* yang lain. Nilai *accuracy* dan *F1-Score* yang didapatkan untuk model DNN dengan 8 *feature* adalah 72% dan 66%, sedangkan untuk model DNN dengan 13 *feature* nilai *accuracy* dan *F1-Score* yang didapatkan adalah 90% dan 88%. Detil dari hasil percobaan dapat dilihat pada Tabel 4 dan Tabel 5.

Tabel 4. Hasil klasifikasi model DNN dengan 8 *feature* dari Dataset

Percobaan DNN model (8 *feature* data NSL-KDD)											
Max_iter / Epoch	Activation	Stopped At	TN	TP	FP	FN	TOTAL	ACCURACY	PRECISION	RECALL	F1-SCORE
500	Tanh	500	9544	1885	167	5575	17171	0,66560	0,91862	0,25268	0,39634
	ReLu	**500**	**7569**	**4814**	**2142**	**2646**	**17171**	**0,72116**	**0,69206**	**0,64531**	**0,66787**
	Sigmoid	103	9711	0	0	7460	17171	0,565547	0	0	0

Tabel 5. Hasil klasifikasi model DNN dengan 13 *feature* dari Dataset

Percobaan DNN model (13 *feature* data NSL-KDD)											
Max_iter / Epoch	Activation	Stopped At	TN	TP	FP	FN	TOTAL	ACCURACY	PRECISION	RECALL	F1-SCORE
500	Tanh	158	9635	5547	76	1913	17171	0,88417	0,98648	0,74357	0,84797
	ReLu	**234**	**9621**	**5958**	**90**	**1502**	**17171**	**0,90729**	**0,98512**	**0,79866**	**0,88214**
	Sigmoid	103	9711	0	0	7460	17171	0,565547	0	0	0

4.3 Pengujian Menggunakan *Signature*-IDS

Hasil perhitungan nilai MTTD dan MTTR untuk setiap simulasi serangan yang dilakukan menggunakan *Signature*-IDS memiliki nilai yang bervariasi. Nilai MTTD dan MTTR yang didapatkan saat simulasi ICMP *flood* adalah 7.2475 detik dan 11.74 detik, lalu untuk simulasi serangan SYN *flood* nilai MTTD yang didapat adalah 26.995 detik dan nilai MTTR adalah 11.00 detik. Kemudian untuk simulasi serangan UDP *flood* didapatkan nilai MTTD sebesar 20.49 detik dan nilai MTTR sebesar 3.00 detik.

4.4 Pengujian Menggunakan *Anomaly*-IDS

Pengujian menggunakan *Anomaly*-IDS yang dilakukan, tidak bisa mendapatkan nilai MTTD dan MTTR. Hal ini disebabkan oleh cara kerja dari *Anomaly*-IDS adalah melakukan klasifikasi per satu *packet*, selain itu juga terdapat *delay* waktu dari *tool kdd99-feature-extractor* yang digunakan. *Delay* waktu yang dihasilkan oleh *tool* tersebut adalah sebesar 60 detik. Adanya *delay* waktu tersebut dikarena adanya proses *sorting* dan *aggregate* oleh *tool* dari *traffic* simulasi yang dijalankan untuk mendapatkan *feature* yang dibutuhkan dan menjadi *data input* model DNN. Hal ini menyebabkan proses klasifikasi yang dilakukan membutuhkan waktu yang lebih lama dari proses simulasi. Kemudian dari kondisi tersebut dipilihlah cara untuk mengevaluasi *Anomaly*-IDS dengan cara menghitung nilai masing – masing elemen dari *confusion matrix* (TP, TN, FP, FN). Dari nilai masing - masing elemen tersebut dihitung tingkat kesalahan dalam klasifisikasi *packet*, hal ini dilakukan dengan alasan saat proses simulasi, model DNN yang melakukan klasifikasi tidak mendapatkan *packet* serangan, sehingga

dihitung jumlah *packet* yang salah diklasifikasikan. Detail dari masing – masing nilai elemen *confusion matrix* berdasarkan variasi serangan dapat dilihat pada Tabel 6 dan Tabel 7.

Tabel 6. Hasil klasifikasi model DNN dengan 8 *feature* dari Simulasi

Percobaan DNN model (8 feature data Simulasi)									
	TN	TP	FP	FN	TOTAL	ACCURACY	PRECISION	RECALL	F1-SCORE
ICMP	215	0	144	7	366	0,58743	0	0	0
SYN	88	36	36	740	900	0,13778	0,50000	0,04639	0,08491
UDP	2641	4	473	2	3120	0,84776	0,00839	0,66667	0,01656

Tabel 7. Hasil klasifikasi model DNN dengan 13 *feature* dari Simulasi

Percobaan DNN model (13 feature data Simulasi)									
	TN	TP	FP	FN	TOTAL	ACCURACY	PRECISION	RECALL	F1-SCORE
ICMP	165	0	283	7	455	0,36264	0	0	0
SYN	55	256	72	557	940	0,33085	0,78049	0,31488	0,44873
UDP	2529	2	585	4	3120	0,81122	0,00341	0,33333	0,00675

5. KESIMPULAN

Berdasarkan hasil pengamatan dan pengujian pada sistem, dapat disimpulkan beberapa hal sebagai berikut:

- Dari beberapa percobaan pembuatan model DNN untuk melakukan klasifikasi / *training* menggunakan *dataset* NSL-KDD menunjukan bahwa, 8 *feature* yang dipilih berdasarkan jurnal rujukan mendapat nilai *accuracy* dan *F1-Score* lebih rendah dibandingkan 13 *feature* hasil *feature selection* dengan metode RFE.
- Simulasi yang dilakukan untuk melakukan deteksi ICMP *flood*, SYN *flood* dan UDP *flood* menggunakan *Signature*-IDS, mendapatkan nilai rata – rata waktu deteksi dan mitigasi lebih cepat / rendah dibandingkan *Anomaly*-IDS. Yaitu untuk mendeteksi serangan ICMP *flood* adalah 7.2475 detik, SYN *flood* adalah 20.995 detik, dan UDP *flood* adalah 20.49 detik. Yaitu untuk melakukan mitigasi serangan ICMP *flood* adalah 11.74 detik, SYN *flood* adalah 11.00 detik, dan UDP *flood* adalah 3.00 detik.
- Simulasi yang dilakukan untuk melakukan deteksi ICMP *flood*, SYN *flood* dan UDP *flood* menggunakan *Anomaly*-IDS, mampu melakukan deteksi serangan DDoS namun membutuhkan waktu yang lebih lama dari waktu yang dibutuhkan oleh *Signature*-IDS. Selain itu hal ini juga membuktikan bahwa *Anomaly*-IDS menggunakan model DNN tidak mampu mendeteksi serangan DDoS yang bersifat *volumetric*, dengan alasan bahwa cara mendeteksi *Anomaly*-IDS lebih fokus pada sifat / karakter per satu packet.
- Serangan ICMP *flood* lebih mudah terdeteksi daripada SYN *flood* dan UDP *flood* saat mennggunakan *Signature*-IDS. Hal ini dikarenakan tidak ada *service port* yang perlu melakukan respon terhadap serangan yang dilakukan. Sedangkan untuk serangan SYN *flood* dan UDP *flood*, server / target perlu untuk mengirimkan RST *packet* ke sumber *packet* (saat terjadi serangan SYN *flood*) dan mengirimkan ICMP *destination port unreachable* (saat terjadi serangan UDP *flood*), sehingga memenuhi dan memperlambat jaringan yang ada.
- Melihat dari nilai *accuracy* yang didapatkan, menujukkan bahwa model DNN dengan 8 *feature* memiliki tingkat ketepatan yang lebih tinggi untuk mengklasifikasikan *packet* ICMP yaitu sebesar 59% dan untuk *packet* UDP sebesar 84%, sedangkan untuk model DNN dengan 13 feature memiliki ketepatan untuk mengklasifikasi *packet* SYN yaitu sebesar 33%.
- Melihat nilai *F1-Score* pada simulasi serangan ICMP *flood* dan UDP *flood* menggunakan *Anomaly*-IDS, nilai yang didapatkan cukup rendah yaitu dibawah 5%. Hal ini disebabkan oleh nilai *precision* dan *recall* yang rendah pula. Sedangkan untuk nilai *F1-Score* pada simulasi SYN *flood* lebih tinggi yaitu 44%. Nilai ini menyatakan bahwa *tool kdd99-feature-extractor* ini kurang tepat untuk digunakan untuk mengklasifikasi *packet* secara simulasi *real-time*. Hal tersebut dibuktikan dengan adanya *delay* waktu untuk *sorting* dan *aggregate* juga jumlah *data packet* yang tidak seimbang yang dikirim ke model DNN (*Anomaly*-IDS).

6. DAFTAR PUSTAKA

[1] Brownlee, J. 2021. How to Choose an Activation Function for Deep Learning. URI= https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/

[2] Brownlee, J. 2021. Tour of Evaluation Metrics for Imbalanced Classification. URI= https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/

[3] Choudhary, S., & Keswani, N. 2020. Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 Datasets using Deep Learning in IoT. 167, 1561-1573. DOI=https://doi.org/10.1016/j.procs.2020.03.367

[4] Craven, C. 2020. What Is Software Defined Networking (SDN)? Definition. URI= https://www.sdxcentral.com/networking/sdn/definitions/what-the-definition-of-software-defined-networking-sdn/

[5] EL-Garoui, L., Pierre, S., & Chamberland, S. 2020. A New SDN-Based Routing Protocol for Improving Delay in Smart City Environments. *3*(3) 1004-1021. DOI=https://doi.org/10.3390/smartcities3030050

[6] Gupta, A., & Sharma, L. S. 2019. Mitigation of DoS and Port Scan Attacks Using Snort. 7(4), 248-258. DOI=https://doi.org/10.26438/ijcse/v7i4.248258

[7] ITExamAnswers Team. 2020. CyberOps Associate: Module 2 – Fighters in the War Against Cybercrime. URI=https://itexamanswers.net/cyberops-associate-module-2-fighters-in-the-war-against-cybercrime.html

[8] Paquet, C. 2009. Network Security Using Cisco IOS IPS. In Implementing Cisco IOS Network Security (IINS): (CCNA Security exam 640-553) (Authorized Self-Study Guide). Cisco Press. URI= https://www.ciscopress.com/articles/article.asp?p=1336425

[9] Rao, U. H., & Nayak, U. 2014. Intrusion Detection and Prevention Systems. In The InfoSec Handbook (pp. 225-243). Apress, Berkeley, CA. DOI=https://doi.org/10.1007/978-1-4302-6383-8_11

[10] Reddy, T. N., & Kumarappan, A. P. 2018. Intrusion Detection on Software Defined Networking. 7, 330-332. DOI=http://dx.doi.org/10.14419/ijet.v7i3.12.16052

[11] Saporito, G. 2019. A Deeper Dive into the NSL-KDD Data Set. URI=https://towardsdatascience.com/a-deeper-dive-into-the-nsl-kdd-data-set-15c753364657

[12] Weisman, S. 2020. What is a distributed denial of service attack (DDoS) and what can you do about them? URI=https://us.norton.com/internetsecurity-emerging-threats-what-is-a-ddos-attack-30sectech-by-norton.html

[13] Western Governors University. 2020. Neural networks and deep learning explained. URI= https://www.wgu.edu/blog/neural-networks-deep-learning-explained2003.html