REPOSITOR, Vol. 2, No. 11, November 2020, Pp. 1491-1502
ISSN : 2714-7975
E-ISSN : 2716-1382

Deteksi Dan Mitigasi Serangan DDoS Pada Software Defined Network Menggunakan Algoritma Decision Tree

Syahputra, Q.*[1], Akbi, D.[2], Risqiwati, D.[3]
[1]Universitas Muhammadiyah Malang
e-mail: 201510370311180@webmail.umm.ac.id*[1], dnarregata@umm.ac.id[2]

Abstrak

Software Defined Network (SDN) merupakan paradigma baru dalam manajemen jaringan yang memberikan fasilitas untuk melakukan konfigurasi, virtualisasi, dan mengolah infrasturktur jaringan secara terpusat. Manajemen jaringan secara terpusat dilakukan pada SDN Controller yang dimana memisahkan network data plane dari control functions. Serangan Distributed Denial of Service (DDoS) adalah salah satu permasalahan besar dalam kemanan jaringan yang menyebabkan services yang ada pada jaringan menjadi tidak dapat diakses dalam jangka waktu tertentu. Penelitian ini bertujuan untuk membuat sistem deteksi menggunakan algortima Decision Tree dan mitigasi serangan DDoS dengan metode drop packet pada Software Defined Network. Model klasifikasi yang telah dibangun berdasarkan dataset CICIDS 2017 diterapkan pada controller dan kemudian menjadi pendeteksi serangan DDoS jenis User Data Protocol (UDP). Setiap packet in yang masuk ke dalam controller akan melalui proses pendeteksian sebelum diteruskan kepada destination source, adapun jika packet in terdeteksi sebagai serangan DDoS maka controller akan melakukan fungsi mitigasi drop packet terhadap host yang terbukti melakukan penyerangan. Dari percobaan yang telah dilakukan UDP Flood terbukti menghabiskan banyak network resources dan meningkatkan penggunaan CPU sehingga menyebabkan controller mengalami gangguan berfungsi selama proses penyerangan berlangsung. Hasil penelitian ini menunjukkan bahwa sistem yang dibuat berhasil melakukan proses deteksi dan mitigasi serangan UDP Flood dengan akurasi sebesar 99.95% dan diikuti proses mitigasi dari setiap paket yang terbukti melakukan penyerangan.

Kata kunci: *SDN, CICIDS 2017, UDP Flood, Decision Tree, Drop Packet.*

Abstract

Software Defined Network (SDN) is a new paradigm in network management that provides facilities to configure, virtualize, and process network infrastructure centrally. Centralized network management is performed on the SDN Controller which separates the network data plane from control functions. Distributed Denial of Service (DDoS) attacks are one of the major problems in network security that cause services on the network to become inaccessible for a certain period of time. This study aims to create a detection system using Decision Tree algorithm and DDoS attack mitigation with the drop packet method on Software Defined Network. The classification model that has been built based on the CICIDS 2017 dataset is applied to the controller and then detects a DDoS attack as type User Data Protocol (UDP). Every packet in that enters in controller will go through to the detection process before it is forwarded to the destination source, while if the packet in is detected as a DDoS attack, the controller will perform the function of drop packet mitigation against the host that is proven to attack. From the experiments that have been carried out UDP Flood proved to consume a lot of network resources and increase CPU usage, causing the controller to experience a malfunction during the attack process. The results of this study indicate that the system created successfully carried out the process of detecting and mitigating UDP Flood attacks with an accuracy of 99.95% and followed by the mitigation process of each package that was proven to carry out attacks.

Keywords*: SDN, CICIDS 2017, UDP Flood, Decision Tree, Drop Packet.*

1. Pendahuluan

Teknologi jaringan dikembangkan oleh berbagai macam vendor dengan kebijakan dari produknya masing-masing dan setiap vendor memiliki kekurangan seperti ketidak- konsistenan

dan kompleksitas penggunaan sehingga mengakibatkan konsumen sulit mengikuti setiap pembaharuan yang dikeluarkan oleh vendor. Perbedaan aturan dan kebijakan teknologi jaringan mengakibatkan permasalahan yang berbeda pula. Oleh karena itu, diciptakanlah suatu arsitektur teknologi jaringan terbaru untuk mencapai satu tujuan yang sama untuk memudahkan pengelolaan jaringan, teknologi jaringan itu dikenal dengan nama SDN (Software Defined Network) [1]. Serangan Distributed Denial of Service (DDoS) menjadi salah satu serangan yang dapat melumpuhkan lalu lintas dan servis jaringan dengan cara membebani server, network link dan perangkat jaringan (switch, router, dll.) dengan traffic network yang sangat tinggi.[2]. Software Defined Network memiliki controller yang memiliki peran yang sangat penting yaitu sebagai pusat akses pengaturan lalu lintas [3]. Dalam perkembangannya beberapa peneliti telah melakukan riset tentang keamanan SDN terhadap serangan DDoS, salah satunya adalah N. Bawany dkk. Yang membahas studi literatur tentang metode pencegahan dan mitigasi terhadap serangan DDoS pada SDN dengan memanfaatkan mininet dan controller SDN yang bersifat open source .

Software Defined Network (SDN) telah muncul yang memungkinkan lebih banyak fleksibilitas melalui kontrol jaringan. Sistem kerja dasar dari SDN adalah memisahkan bidang kontrol dari bidang data menjadi sebuah program, yang disebut controller. Controller bersifat Programable untuk melakukan pengaturan komponen jaringan secara terpusat seperti pembaharuan tabel jaringan , control traffic dan pengaturan kebijakan lainnya. Keamanan telah menjadi perhatian penting dari arsitektur jaringan SDN karena sejak pertama kali terbentuknya, teknologi ini belum memiliki fitur keamanan bawaan hingga pada akhirnya para peneliti melakukan riset lebih lanjut untuk mengembangkan sistem keamanan dan efisiensi pengiriman paket data. Penelitian telah menunjukkan bahwa berbagai serangan keamanan dapat dilakukan terhadap sistem SDN melalui berbagai komponen jaringan seperti perangkat lunak, pembentukan topologi dan kerentanan kode juga memiliki dampak penting pada keamanan SDN. Apalagi SDN menawarkan banyak peluang untuk menerapkan kontrol keamanan secara dinamis dan virtual [4] .

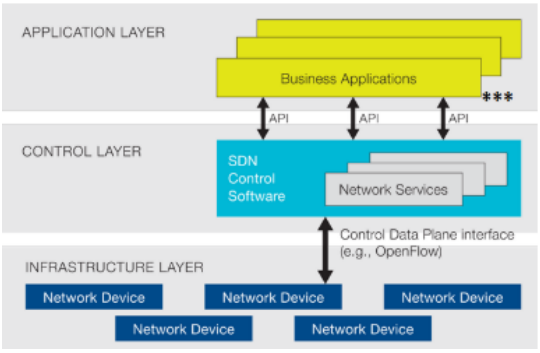
Deteksi serangan DDoS dalam suatu jaringan memiliki beberapa metode dan berbagai macam algoritma, R. Kokila, dkk. telah melakukan penelitian tentang deteksi dan analisis serangan DDoS pada lingkungan SDN menngunakan algoritma klasifikasi SVM (Support Vektor Machine). Kebutuhan penelitian dilakukan dalam ruang lingkup virtual yang dengan kebutuhan perangkat virtual switch support Openflow, dan menggunakan data training dari NSL-KDD untuk membentuk pola klasifikasi. Hasil dari penelitian R. Kokila, dkk. Menemukan bahwa akurasi dari pendeteksian serangan sebesar 95.11% dan ini sudah cukup membuktikan jika algoritma dalam machine learning lainnya dapat diterapkan dalam penelitian yang akan dilakukan [5]. Pemilihan metode machine learning dikarenakan proses komputasi yang cepat, akurasi yang tinggi, dan juga bisa melakukan proses komputasi dengan data yang sanagat banyak. Oleh sebab itu, sangat penting untuk merealisasikan sistem deteksi dan mitigasi DDoS untuk menjaga keamanan jaringan. Dalam penelitian ini akan dilakukan deteksi serangan DDoS pada Software Defined Network menggunakan algoritma Decision Tree.

2. Metode Penelitian

Berikut adalah gambar langkah-langkah penelitian yang akan dilakukan berdasarkan gambar 2.1 dimulai dengan langkah studi literatur, analisis kebutuhan sistem dan penggalian kebutuhan data, perancangan dan pembuatan sistem, pengujian dan dokumentasi, Penrikan kesimulan dan saran.

2.1. Software Defined Network

Konsep dasar dari SDN adalah pemisahan antara control plane dan data plane yang memberikan kemudahan dalam pengelolaan, desain, dan implementasi jaringan secara terpusat dan tidak perlu melakukan banyak konfigurasi di setiap perangkat nm jaringan. Karakteristik SDN menggabungkan seluruh control plane menjadi sebuah controller yang bersifat programmable software sehingga dapat mengontrol banyak perangkat sekaligus dalam sebuah data plane. Sifat sentralisasi ini mempermudah dalam pengoperasian dan pemeliharaan jaringan secara keseluruhan [6]. Gambar 1 di bawah ini menjelaskan arsitektur SDN.

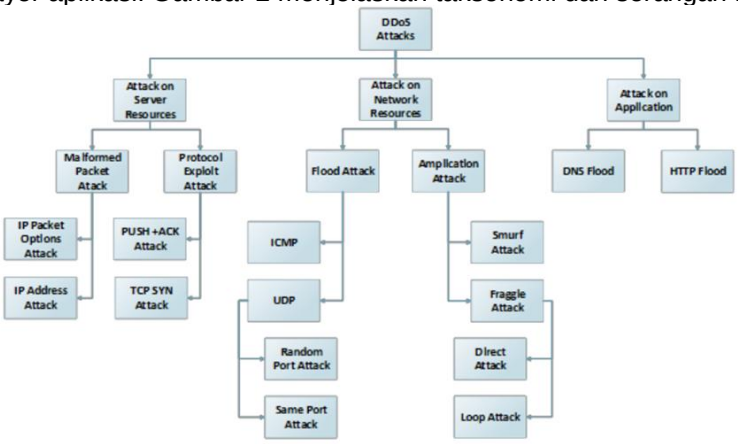


Gambar 1 Arsitektur SDN [11]

Dapat disimpulkan bahwa control layer memegang kendali penuh untuk mengontrol seluruh device yang terkoneksi terhadap SDN control software. Control layer juga dengan bebas melakukan pengaturan API pada application layer dan meneruskan aplikasi tersebut kepada device secara keseluruhan atau spesifik.

2.2. Distributed Denial of Service

Tujuan utama dari serangan DDoS adalah melumpuhkan servis yang ada pada target serangan menggunakan banyak source yang bersifat terdistribusi. Dalam pengimplementasiannya serangan ini memiliki banyak cara untuk melumpuhkan servis salah satunya adalah dengan mengirimkan paket dalam jumlah yang besar secara terdistribusi ke perangkat target sehingga mengakibatkan antrian paket yang berlebihan pada perangkat, hal ini selanjutnya dikenal dengan teknik flooding. Untuk memunculkan efek yang signifikan terhadap target, pelaku penyerangan menggunakan botnets untuk memperbanyak jumlah pelaku serangan (host). Jadi seolah-olah pelaku serangan DDoS menjadi lebih dari satu dan sulit untuk mengidentifikasi pelaku penyerangan yang asli [7]. Dalam pengelompokannya DDoS dapat menyerang beberapa bagian utama dalam suatu sistem. Serangan server resources, network resources, dan layer aplikasi. Gambar 2 menjelaskan taksonomi dari serangan DDoS.



Gambar 2. Taksonomi Serangan DDoS [2]

Tipe serangan DDoS ini bersifat connectionless yang menghabiskan network resource dari korban seperti bandwidth dengan cara membanjiri perangkat jaringan menggunakan paket User Data Protocol (UDP) Flood dalam jumlah yang banyak. Serangan ini menggunakan IP address secara acak dalam jumlah banyak atau spesifik dan menyerang port secara acak atau spesifik [2]. Serangan UDP flood memiliki beberapa karakteristik sebagai berikut: (1) jumlah serangan dalam skala besar dan dilakukan dengan rentan waktu tertentu, (2) Penyerang sering diidentifikasi sebagai pengguna jaringan yang sah, (3) Pola serangan akan bercampur untuk memicu serangan yang nyata. Berdasarkan karakteristik yang telah dijabarkan di atas, berikut adalah parameter dari UDP Flood yang ditunjukkan pada tabel 1. [8].

Tabel 1. Parameter UDP Flood

Time	*Source*	*Destination*	*Protocol*	*Dest. Port*	*Length/bytes*

Parameter didasari oleh waktu pengiriman yang selanjutnya diikuti oleh Source atau alamat pengirim. Kemudian paket akan dikirimkan ke IP address penerima (Destination)

menggunakan Protocol tertentu kepada port tujuan. Parameter di atas menjadi acuan pada algoritma Decision Tree untuk membentuk pola klasifikasi.

2.3. Kebutuhan Perangkat Lunak dan Keras

Tools yang digunakan dalam penelitian ini adalah mininet, Ryu SDN framework, mininet, dan wireshark dengan spesifikasi sebagai berikut :

Tabel 2. Spesifikasi Pserangkat Lunak

Kebutuhan	Versi
SDN Framework/Controller	RYU 4.30
Simulator Jaringan	Mininet versi 2.2.2
Pembuat Paket Serangan	Scapy versi 2.4.2
Machine Learning	Scikit-learn versi 0.20.2
Monitoring	Wireshark versi 2.6.6
Pengiriman Paket Serangan	Tcpreplay

Kebutuhan perangkat keras untuk membangun sistem SDN adalah laptop dengan spesifikasi sebagai berikut.

Tabel 3. Spesifikasi Perangkat Lunak

Nama	Spesifikasi
Sistem Operasi	Ubuntu 18.04
Processor	Intel(R) Core(TM) i3 CPU M350 @ 2.27GHz
RAM	2 GB
System type	64-bit *Operating System*, x64-*based processor*

2.4. Algortima Pendeteksian

Decision Tree adalah metode pengklasifikasian yang menggunakan fungsi-fungsi pendekatan diskrit [1]. Terdapat banyak algoritma Decision Tree seperti ID3, C4.5, C5.0, dan CART. Pada dasarnya algoritma Decision Tree menggunakan perhitungan enthropy kemudian information gain untuk membentuk pohon keputusan. Pemilihan atribut diproses menggunakan information gain. Berikut adalah formula perhitungan nilai entropy dan information gain [9].

$$I\left(S_1, S_{2,} S_{3,\dots,} S_m\right) = -\sum_{i=1}^{m} p_i log_2(p_i) \quad (1)$$

S = Himpunan
m = Banyaknya kelas
p_i = Proporsi kelas

Untuk memperoleh informasi nilai subset dari A dan seterusnya digunakan formula

$$E(A) = \sum_{j=1}^{y} \frac{S_{1j} + S_{2j} + \dots S + mj}{S} \; I\left(S_{1j}, S_{2j}, \dots, S_{mj}\right) \quad (2)$$

S_1j = Sampel kelas i
j = Atribut A
E = Entropy

Setelah semua perhitungan Entropy total dan subset selesai selanjutnya akan dilakukan perhitungan nilai Information gain denga formula :

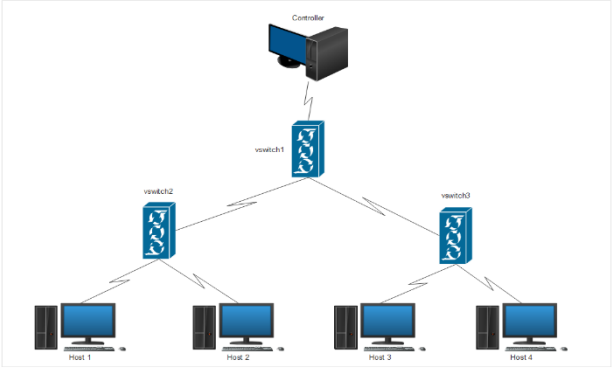
$$GAIN\ (A) = I(S_1, S_2, \dots, S_{1m}) - E(A) \quad (3)$$

Pada penelitian ini menggunakan Decision Tree sebagai landasan algoritma dalam proses pengklasifikasian serangan UDP Flood. Pembentukan pohon keputusan algoritma CART terdiri dari beberapa tahap (1) menentukan node akar dengan menghitung nilai information gain dari setiap atribut. Nilai information gain tertinggi akan menjadi node akar, (2) pembentukan node

cabang berdasarkan nilai information gain tertinggi kedua dan seterusnya, (3) langkah ini dilakukan secara terus-menerus sampai nilai information gain terakhir telah ditemukan [9].

2.5. Topologi Jaringan

Berikut adalah gambar 3 perancangan topologi jaringan SDN.

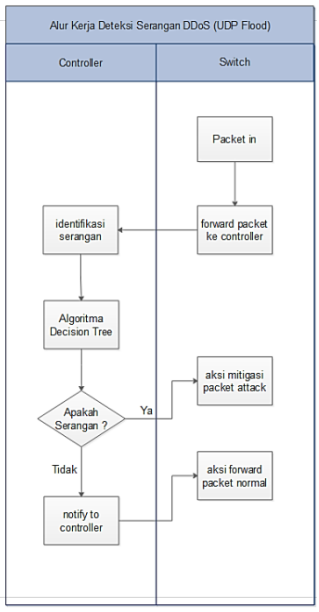


Gambar 3. Rancangan Topologi Jaringan

Topologi simulasi mininet di atas terdiri dari satu controller (Ubuntu), tiga OVSKernel switch openflow, dan 4 host yang terhubung dengan OVSKernel switch. Topologi inilah yang akan digunakan dalam simulasi penyerangan pada lingkungan virtual. Karena keterbatasan sumber daya perangkat keras maka topologi mininet diperlukan untuk membuat prototype jaringan kecil atau besar yang bersifat virtual [10].

2.6. Alur Kerja Deteksi DDoS

Berikut adalah workflow (alur kerja) dari sistem deteksi DDoS :



Gambar 4. Alur Kerja Deteksi DDoS

Paket yang dikirimkan oleh source host akan melalui switch sebelum diteruskan ke host tujuan, setelah paket telah sampai ke switch akan dikirim ke controller. Selanjutnya akan dilakukan proses pendeteksian serangan menggunakan algoritma Decision Tree yang telah diimplementasikan ke dalam controller. Hasil dari proses pengklasifikasian terdiri dari 2 kelas yaitu UDP flood traffic diwakilkan dengan angka 1 dan normal traffic dengan angka 0, jika paket yang dikirimkan terbukti sebagai serangan maka controller akan segera m elakukan metode mitigasi block packet pada source yang terbukti melakukan serangan. Jika serangan dikategorikan sebagai normal traffic maka controller akan melakukan konfirmasi ke switch dan meneruskan paket ke tujuan.

2.7. Paket Serangan DDoS (UDP Flood)

Paket serangan DDoS UDP flood dibuat menggunakan scapy dengan bahasa pemrograman python memanfaatkan library yang sudah ada di dalamnya. Tabel 4 menjelaskan spesifikasi paket serangan.

Tabel 4. menjelaskan spesifikasi paket serangan.

Nama	Keterangan
Protokol	UDP
Source IP, Dest. IP, dan Port	Sesuai dengan dataset *learning*
Ethernet type	0x800 (ipv4)
Interface	h1-eth0

Berdasarkan penelitian [3] yang menganalisis dampak serangan DDoS tipe TCP dan UDP flood menunjukkan penggunaan sumber daya (CPU meningkat menjadi +-42% dan memory menjadi +-18%) pada layer controller saat serangan dilancarkan. Penelitian tersebut melakukan pengujian pengiriman paket serangan selama 30 detik dengan paket perdetiknya masing-masing berjumlah 1000, 3000, hingga 5000 paket/detik. Karena keterbatasan jumlah serangan UDP Flood pada dataset maka berikut rincian jumlah paket UDP yang akan digunakan dalam pengujian serangan

Tabel 5. Paket Serangan UDP flood

Paket (per detik)	Jumlah Waktu (detik)	Total Paket
100	10	1000
200	10	2000
300	10	3000
400	10	4000

2.8 Mitigasi Serangan DDoS (UDP Flood)

Pada bagian ini dijelaskan bahwa kode program pendeteksian dijalankan pada ryu controller. Ryu menyediakan beberapa contoh kode yang bertujuan agar para pengembang dapat menggunakannya sebagai sumber penelitian. Pada penelitian ini penulis memanfaatkan file simpleswitch13 yang sudah ada sebelumnya dan menambahkan dua fitur yaitu deteksi dan mitigasi serangan. Proses pendeteksian dengan memanfaatkan algoritma Decision Tree (machine learning). Proses mitigasi serangan dengan mengatur priority flow rule untuk melakukan aksi drop port. Sumber kode di bawah menjelaskan tentang alur kerja algoritma aturan mitigasi serangan [11].

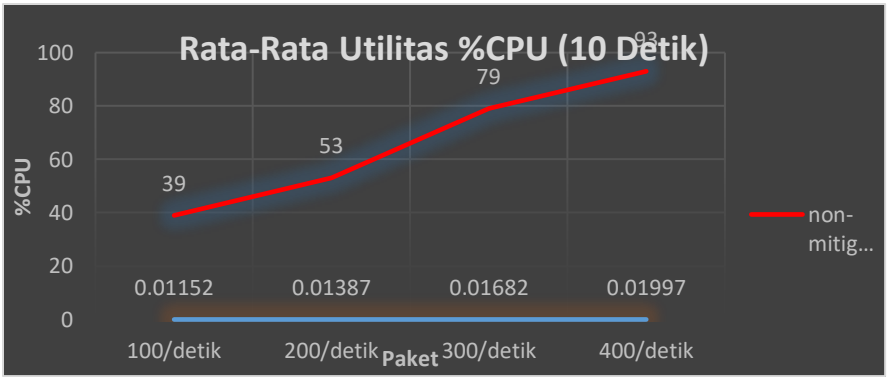
```
if 'ipv4' in header_list:
            self.i = self.i + 1
            ipv4s = pkt.get_protocols(ipv4.ipv4)[0]
            ipsrc = ipv4s.src
            ipdst = ipv4s.dst
            proto = ipv4s.proto
            #srcport = 0
            #dstport = 0

            if 'udp' in header_list:
                udps = pkt.get_protocols(udp.udp)[0]
                srcport = udps.src_port
                dstport = udps.dst_port
                length  = udps.total_length
                print('--------------------------------------')
                print(packet.Packet(msg.data))
                print('--------------------------------------')
                try:
                    y_pred  =self.machine_learning(ipsrc,  ipdst,  srcport,
                              dstport, proto, length)
                    if y_pred[0] == 1:
                        match=              parser.OFPMatch(eth_type=0x0800,
                          in_port=in_port, ip_proto=proto)
```

```
                    Inst=parser.OFPInstructionActions(ofproto.OFPIT_CL
                    EAR_ACTIONS, [])]
                     mod=parser.OFPFlowMod(datapath=datapath,buffer_id
                     =msg.buffer_id,priority=60
                     match=match,instructions=inst,hard_timeout=10)dat
                     apath.send_msg(mod)
                     print("DDoS Terdeteksi, Nilai Prediksi", y_pred)
                return
            elif y_pred[0] == 0:
                print("Paket Normal, Nilai Prediksi : ",y_pred)

        except Exception as e:
            print("error")
            match = parser.OFPMatch(eth_type=0x0800, in_port=in_port,
                        ip_proto=proto)

              inst=[parser.OFPInstructionActions(ofproto.OFPIT_CLEAR
              _ACTIONS, [])]
               mod       =       parser.OFPFlowMod(datapath=datapath,
               buffer_id=msg.buffer_id,priority=60,
               match=match,instructions=inst, hard_timeout=10)
            datapath.send_msg(mod)
            #print(y_pred[0])
            return
```

3. Hasil Penelitian dan Pembahasan

Pada bagian ini akan menampilkan hasil dari lima skenario pengujian yaitu, utilitas CPU controller, packet in dan out, waktu proses dan mitigasi serangan DDoS, quality of service, dan pengaruh proses mitigasi serangan terhadap throughput dan jitter.

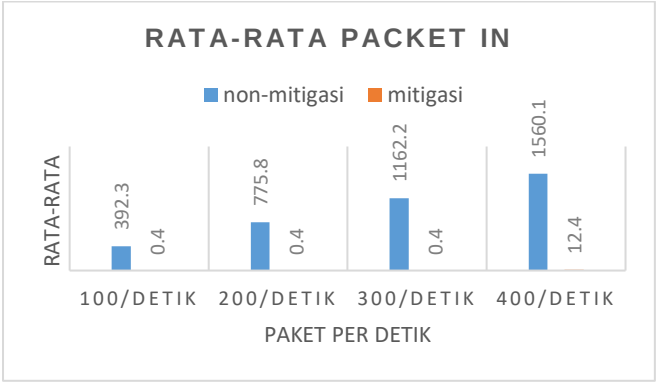
3.1. Utilitas CPU



Gambar 5. Utilitas CPU Controller

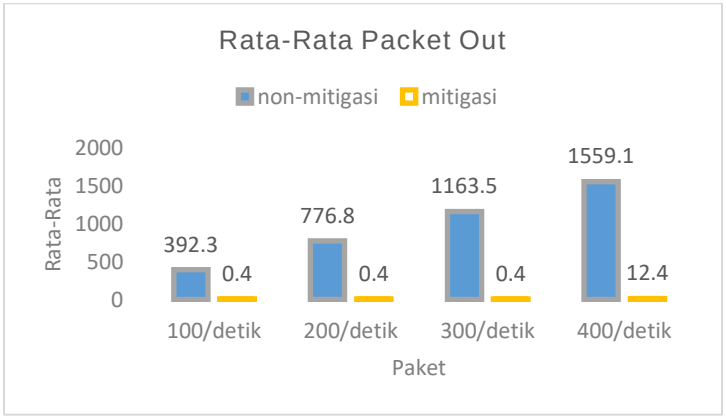
Utilitas CPU menjadi lebih tinggi jika sistem deteksi+mitigasi tidak terpasang pada sistem. Hal ini diakibatkan oleh banyaknya proses antrian paket UDP. Dan jika kita lihat peningkatan CPU berbanding lurus dengan jumlah paket per detik yang dikirimkan.

3.2. Packet In dan Out



Gambar 6. Rata-Rata Packet In

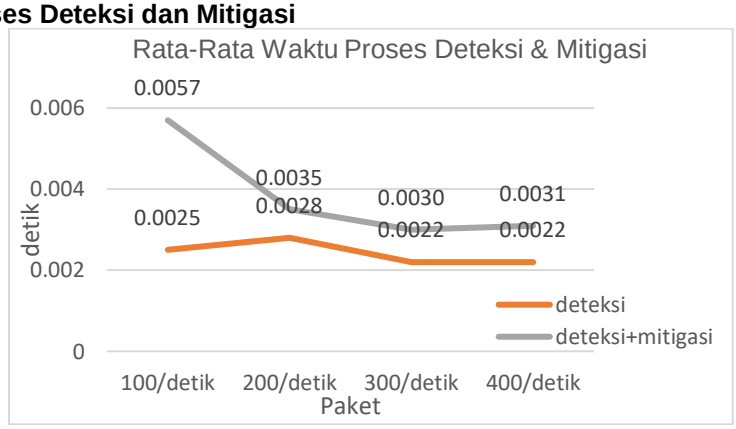
Packet in pada gambar diatas menunjukkan selama proses UDP Flood dilakukan, ketika controller tidak terpasang aplikasi deteksi dan mitigasi, maka packet DDoS sampai pada controller danterhitung sebagai packet in. Setelah aplikasi deteksi dan mitigasi terpasang rata rata packet in menjadi menurun drastis. Dengan ini dapat dikatakan aplikasi deteksi dan mitigasi berjalan dengan baik. Packet out pada gambar 3.2 di bawah menunjukkan selama proses UDP Flood dilakukan, ketika controller terpasang aplikasi deteksi dan mitigasi, maka packet DDoS tidak diteruskan kepada victim, rata rata packet out menjadi menurun drastis. Dengan ini dapat dikatakan aplikasi deteksi dan mitigasi berjalan dengan baik. Selanjutnya akan dilakukan anilasi Quality of Service.



Gambar 7. Rata-Rata Packet Out

Packet out pada gambar diatas menunjukkan selama proses UDP Flood dilakukan, ketika controller terpasang aplikasi deteksi dan mitigasi, maka packet DDoS tidak diteruskan kepada victim, rata rata packet out menjadi menurun drastis. Dengan ini dapat dikatakan aplikasi deteksi dan mitigasi berjalan dengan baik. Selanjutnya akan dilakukan anilasi Quality of Service.

3.3. Waktu Proses Deteksi dan Mitigasi



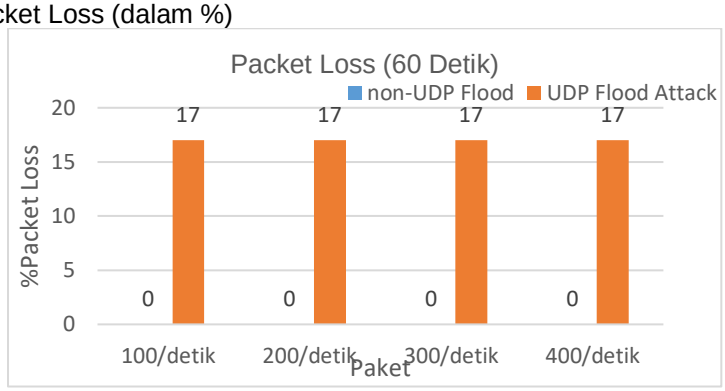
Gambar 8. Waktu Proses Deteksi dan Mitigasi

Jika sistem hanya menjalankan proses deteksi saja tanpa mengaktifkan fungsi mitigasi maka waktu proses yang dibutuhkan lebih sedikit. Akan tetapi, jika aplikasi mitigasi juga dijalankan (deteksi+mitigasi) maka sistem akan membutuhkan waktu lebih banyak untuk melakukan proses.

3.4. Analisa Quality of Service

Pada bagian ini peneliti akan membahas kualitas jaringan sebelum dan pada saat serangan DDoS sedang berlangsung. Analisa QoS terdiri dari Packet Loss, Jitter, dan Throughput.

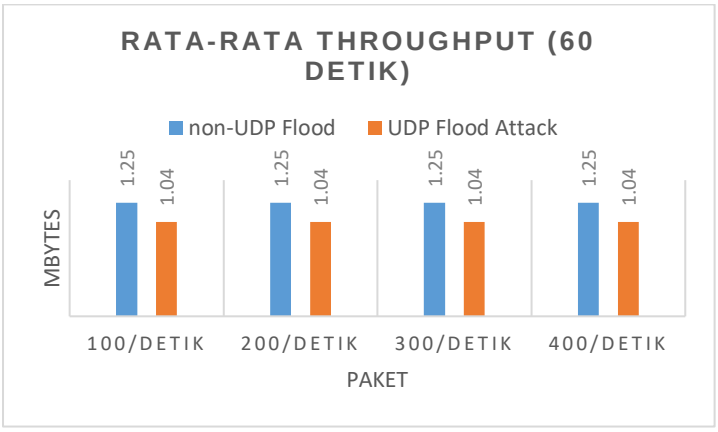
3.4.1 Jumlah Packet Loss (dalam %)



Gambar 9. Packet Loss

UDP Flood mengalami packet loss dikarenakan 17% packet tersebut telah termitigasi dengan metode block port, metode mitigasi tersebut adalah drop seluruh packet yang dikirimkan oleh h1 melalui port dimana dia terkoneksi pada switch selama 10 detik. Setelah 10 detik port akan dibuka lagi untuk h1 dan melakukan proses deteksi kembali.

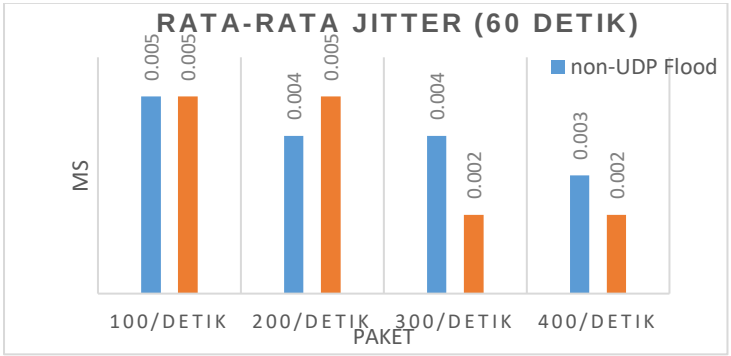
3.4.2 Throughput



Gambar 10. Throughput

Proses analisa throughput dilakukan selama 60 detik dengan durasi penyerangan 10 detik tiap masing masing jenis paket serangan. Gambar 3.6 menunjukkan bahwa serangan DDoS mengurangi nilai Throughput meskipun serangan hanya dilakukan selama 10 detik

3.4.3 Jitter

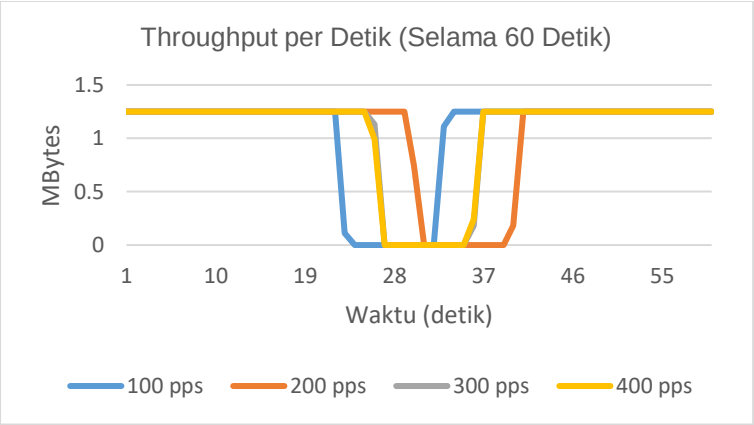


Gambar 11. Rata-Rata Jitter

Proses analisa rata-rata jitter dilakukan selama 60 detik dengan durasi penyerangan DDoS selama 10 detik tiap masing masing jenis paket serangan. Gambar 3.7 menunjukkan bahwa serangan DDoS mengurangi nilai Jitter meskipun serangan hanya dilakukan selama 10 detik.

3.5. Pengaruh Proses Mitigasi terhadap Throughput dan Jitter

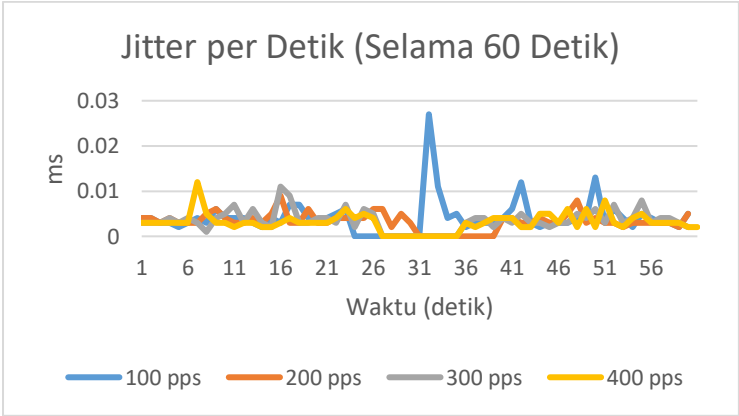
3.5.1. Throughput



Gambar 12. Pengaruh Mitigasi Terhadap Throughput

a. 100pps : serangan terjadi pada detik ke-23 dan berakhir pada detik ke-33, proses berjalan berjalan selama 10 detik sesuai dengan skenario
b. 200pps : serangan terjadi pada detik ke-30 dan berakhir pada detik ke-40, proses berjalan berjalan selama 10 detik sesuai dengan skenario
c. 300pps : serangan terjadi pada detik ke-26 dan berakhir pada detik ke-36, proses berjalan berjalan selama 10 detik sesuai dengan skenario
d. 400pps : serangan terjadi pada detik ke-26 dan berakhir pada detik ke-36, proses berjalan berjalan selama 10 detik sesuai dengan skenario

3.5.2. Jitter



Gambar 13 Pengaruh Mitigasi Terhadap Jitter

a. 100pps : serangan terjadi pada detik ke-23 dan berakhir pada detik ke-33, proses berjalan berjalan selama 10 detik sesuai dengan skenario
b. 200pps : serangan terjadi pada detik ke-30 dan berakhir pada detik ke-40, proses berjalan berjalan selama 10 detik sesuai dengan skenario
c. 300pps : serangan terjadi pada detik ke-26 dan berakhir pada detik ke-36, proses berjalan berjalan selama 10 detik sesuai dengan skenario
d. 400pps : serangan terjadi pada detik ke-26 dan berakhir pada detik ke-36, proses berjalan berjalan selama 10 detik sesuai dengan skenario

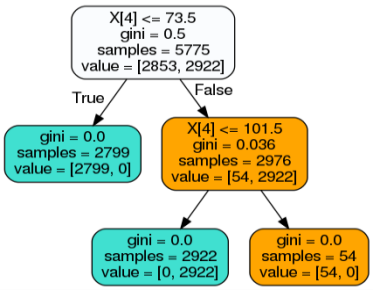
3.6 Akurasi Pendeteksi Serangan DDoS

Tabel 6 menjelaskan hasil dari proses pembuatan model prediksi didapatkan hasil evaluasi

Tabel 6. Evaluasi Algoritma Pendeteksian

Jenis Evaluasi	Nilai (%)
Akurasi	99, 95
Recall	100
Precision	99,91
F1-Score	99.95

Berikut adalah gambar pohon keputusan yang dibuat dengan cara memvisualisasikan graph dari classifier():



Gambar 14. Visualisasi Pohon Keputusan

4. Kesimpulan

a. Dalam penelitian ini berhasil menbangun sistem deteksi dan mitigasi serangan DDoS yang dibuktikan dengan berbagai skenario pengujian.
b. Pembentukan model pendeteksian (rule detection) bergantung kepada dataset acuan, yang kemudian dilakukan proses learning untuk mendapatkan pola serangan dan dibentuk menjadi pohon keputusan untuk melabeli setiap paket yang masuk.
c. UDP Flood menjadi fokus utama dalam proses deteksi dan mitigasi. Karena jenis serangan DDoS ini sulit untuk terindentifikasi karena sifat connectionless-nya dan memakan banyak resources dalam jaringan. Akan tetapi, karena keterbatasan dataset yang dimiliki maka peneliti dalam hal ini hanya mendapatkan delapan ribu baris data (empat ribu benign label dan 4 ribu attack label) yang dijadikan dataset learning.
d. Model pendeteksian yang dibangun melalui proses learning mendapatkan akurasi sebesar 99.95 % dan dibuktikan dengan simulasi serangan dan pengujian.
e. Metode mitigasi yang digunakan adalah mitigasi block port dengan kebijakan menutup port yang positif melakukan penyerangan selama 10 detik. Setelah 10 detik port akan dibuka kembali dan melakukan proses deteksi dan mitigasi kembali.
f. Topologi yang digunakan adalah 3 switch dan 4 host. Dengan topologi demikian UDP Flood bisa menghabiskan banyak resources seperti CPU utilization, QoS, dan penggunaan memory perangkat keras.

5. Saran

Terbatasnya jumlah dataset UDP Flood menjadi salah satu kendala dalam pembentukan model pendeteksian akan lebih baik semakin banyak jumlah dataset maka model pendeteksian akan semakin kuat dan akurat. Di sisi lain, dengan jumlah dataset yang terbatas hasil yang didapatkan melalui eksperimen dan skenario pengujian menunjukkan bahwa sistem yang dibangun berfungsi dengan baik. Pada penelitian ini terbatas pada lingkungan virtual saja dengan memanfaatkan mininet sebagai simulator jaringan. Tentunya proses konfigurasi SDN antara virtual dan perangkat keras nyata memiliki perbedaan. Diharapkan pada penelitian berikutnya dapat menerapkan sistem deteksi dan mitigasi serangan DDoS pada SDN pada perangkat keras nyata dengan menggunakan metode dan langkah teknis yang telah dijelaskan. Sehingga, penelitian ini benar-benar tervalidasi secara teori dan teknis sehingga dapat diimplementasikan ke dalam dunia nyata.

Refrensi

[1] S. R. Afif, P. Sukarno, and M. A. Nugroho, “Analisis Perbandingan Algoritma Naive Bayes dan Decision Tree untuk Deteksi Serangan Denial of Service (DoS) pada Arsitektur Software Defined Network (SDN) Pendahuluan Studi Terkait,” *e-Proceeding Eng.*, vol. 5, no. 3, pp. 7515–7521, 2018.
[2] A. P. Fajar and T. W. Purboyo, “A Survey Paper of Distributed Denial-of-Service Attack in Software Defined Networking (SDN),” *Int. J. Appl. Eng. Res. ISSN*, vol. 13, no. 1, pp. 973–4562, 2018.
[3] S. Sondur, “Software Defined Networking for Beginners Software Defined Networking for Beginners,” *Temple Univ.*, vol. 1, no. September, pp. 1–9, 2015.
[4] I. Alsmadi and D. Xu, “Security of Software Defined Networks: A survey,” *Comput. Secur.*,

vol. 53, pp. 79–108, 2015.

[5] R. T. Kokila, S. Thamarai Selvi, and K. Govindarajan, “DDoS detection and analysis in SDN-based environment using support vector machine classifier,” *6th Int. Conf. Adv. Comput. ICoAC 2014*, pp. 205–210, 2015.

[6] Y. S. H. Roni Fernando Simarmata, Rohmat Tulloh, “SIMULASI JARINGAN SOFTWARE DEFINED NETWORK MENGGUNAKAN PROTOKOL ROUTING OSPF DAN RYU CONTROLLER,” *e-Proceeding Appl. Sci.*, vol. 4, no. 3, pp. 2887–2896, 2018.

[7] N. Z. Bawany, J. A. Shamsi, and K. Salah, “DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions,” *Arab. J. Sci. Eng.*, vol. 42, no. 2, pp. 425–441, 2017.

[8] K. Ramadhani, M. Yusuf, H. E. Wahanani, J. T. Informatika, and F. T. Industri, “Anomali Perubahan Traffic Jaringan Berbasis Cusum,” *Comput. Secur.*, pp. 1–9, 2013.

[9] S. Guntur Arief Darmawan, Yuliana Susanti, “IMPLEMENTASI DATA MINING MENGGUNAKAN ALGORITME C5.0 PADA DATA KELULUSAN MAHASISWA S1 UNIVERSITAS SEBELAS MARET SURAKARTA,” *Univ. Sebel. Maret*, pp. 1–8, 2018.

[10] R. Kartadie, E. Utami, and E. Pramono, “Prototipe Infrastruktur Software-Defined Network Dengan Protokol Openflow Menggunakan Ubuntu Sebagai Kontroler,” *Dasi*, vol. 15, no. 1, pp. 24–32, 2014.

[11] U. M. Gurusamy, K. Hariharan, and M. Msk, “Detection and mitigation of UDP flooding attack in a multicontroller software defined network using secure flow management model,” *Concurr. Comput.* , no. February, pp. 1–11, 2019.