SISTEM DETEKSI SERANGAN DDoS PADA *SOFTWARE DEFINED NETWORK* MENGGUNAKAN METODE RANDOM FOREST

1st Pranata Alam
Universitas Telkom
S1 Sistem Informasi
Bandung, Indonesia
pranataalam@student.telkomuniversity.ac.id

2nd Dr. Mochammad Teguh Kurniawan, S.T., M.T
Universitas Telkom
S1 Sistem Informasi
Bandung, Indonesia
teguhkurniawan@telkomuniversity.ac.id

3rd Umar Yunan KS Hediyanto, S.T., M.T
Universitas Telkom
S1 Sistem Informasi
Bandung, Indonesia
umaryunan@telkomuniversity.ac.id

Abstrak*— Pada era digital yang semakin kompleks, serangan siber jenis Distributed Denial of Service (DDoS) telah menjadi ancaman serius bagi keamanan jaringan, karena mampu melumpuhkan sistem dan layanan dengan membanjiri target menggunakan lalu lintas data yang berlebihan. Penelitian ini mengusulkan sistem deteksi serangan DDoS inovatif yang menggabungkan kekuatan jaringan yang terdefinisi perangkat lunak (Software-Defined Network/SDN) dengan algoritma pembelajaran mesin Random Forest. Data lalu lintas dianalisis untuk menghasilkan fitur-fitur seperti jumlah paket, waktu antar paket, dan ukuran rata-rata paket. Fitur-fitur ini digunakan karena dikenal memiliki kemampuan diskriminatif tinggi dalam mendeteksi pola anomali. Selanjutnya, model yang telah dilatih diintegrasikan ke dalam kontroler SDN, seperti Ryu Controller, sehingga mampu mendeteksi pola lalu lintas mencurigakan secara cepat dan akurat. sistem ini menunjukkan efektivitasnya dengan tingkat akurasi deteksi sebesar 88%. Hasil ini menunjukkan bahwa kombinasi teknologi SDN dan Random Forest memberikan solusi yang andal dalam melindungi infrastruktur digital dari serangan DDoS, sehingga dapat mencegah kerusakan signifikan pada sistem jaringan..

Kata kunci*— Software Defined Network (SDN), Distributed Denial of Service (DDoS), Random Forest.

I. PENDAHULUAN

Seiring berjalannya waktu, berkembanglah sebuah teknologi manajemen jaringan dengan pendekatan dimana administrator mampu untuk mengontrol trafik pada jaringan secara cloud atau remote. Arsitetktur jaringan tersebut disebut Software Defined Network (SDN) atau istilahnya "Jaringan yang diatur oleh Software". Software Defined Network (SDN) telah menjadi tulang punggung bagi organisasi yang menginginkan fleksibilitas dan efisiensi dalam manajemen jaringan. SDN memungkinkan pemisahan antara kontrol dan data plane, memungkinkan pengelolaan sumber daya jaringan dengan lebih dinamis. Meskipun memberikan keunggulan ini, SDN juga membuka celah baru untuk serangan, dengan serangan Distributed Denial of Service (DDoS) menjadi salah satu ancaman utama.

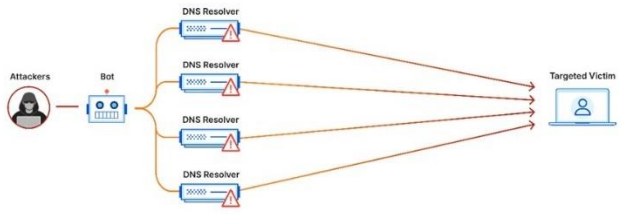
Serangan Distributed Denial of Service (DDoS) merupakan salah satu ancaman utama dalam dunia siber. Serangan ini bertujuan untuk mengganggu layanan jaringan dengan cara menghambat akses pengguna yang dapat menyebabkan kerugian finansial dan reputasi. Serangan DDoS saat ini semakin canggih dan sulit dideteksi karena memiliki bentuk, intensitas, ataupun sumber yang bervariasi. Dengan serangan yang terus berkembang, maka mengharuskan penggunaan solusi deteksi yang lebih tidak kalah cerdas.

Sebagai langkah mengatasi ancaman ini, terdapat beberapa metode mitigasi yang dapat dilakukan untuk mengurangi risiko DDoS, salah satunya adalah penerapan Machine Learning (ML). Algoritma Klasifikasi Random Forest merupakan salah satu metode ML yang potensial untuk membuat sistem deteksi intrusi yang andal. Pada penerapan Machine Learning, dilakukan pelatihan data (train dataset) menggunakan trafik normal (normal traffic) dan trafik serangan DDoS (attack traffic). Dengan memanfaatkan sFlow sebagai mekanisme pemantauan trafik secara realtime, algoritma Random Forest diterapkan untuk melakukan klasifikasi trafik dan mendeteksi keberadaan serangan DDoS secara lebih akurat.

II. KAJIAN TEORI

II.1 Distributed Denial of Service (DDoS)

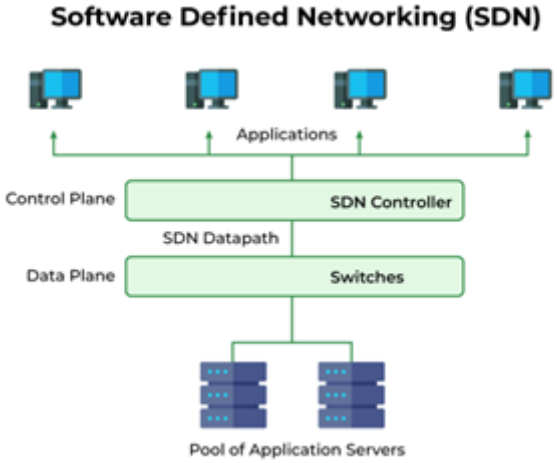
Distributed Denial of Service (DDoS) merupakan serangan siber yang pelakunya berusaha membuat mesin atau sumber daya jaringan tidak tersedia bagi pengguna yang dituju dengan mengganggu layanan host yang terhubung ke jaringan untuk sementara atau tanpa batas waktu. Penolakan layanan biasanya dilakukan dengan membanjiri mesin atau sumber daya yang ditargetkan dengan permintaan yang berlebihan dalam upaya membebani sistem secara berlebihan dan mencegah terpenuhinya beberapa atau semua permintaan yang sah.



Gambar II.1 Distributed Denial of Service

II.2 Software Defined Network (SDN)

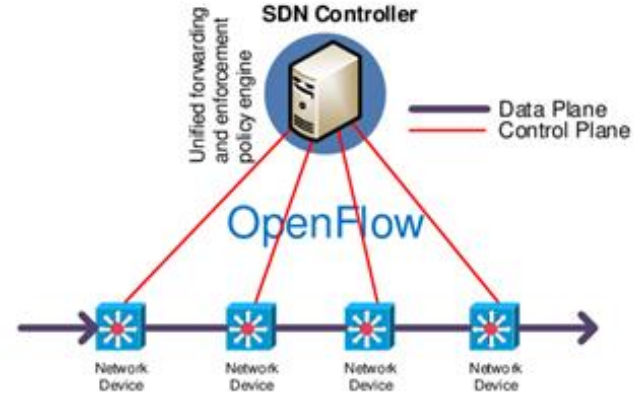
Software Defined Networking (SDN) adalah pendekatan revolusioner dalam arsitektur jaringan komputer modern. Dengan SDN, kontrol atas infrastruktur jaringan dipisahkan dari perangkat kerasnya dan dipusatkan di satu lokasi. Pendekatan ini membuka pintu bagi cara-cara baru dalam mengelola jaringan, termasuk integrasi aplikasi pihak ketiga, konfigurasi yang dinamis dan adaptif, serta dukungan untuk layanan berbasis awan. (Tupakula, Karmakar, 2020). SDN adalah pendekatan inovatif untuk merancang, mengimplementasikan, dan mengelola jaringan yang memisahkan kontrol jaringan (bidang kendali) dan proses penerusan (bidang data) untuk pengalaman pengguna yang lebih baik.



Gambar II.2 Software Defined Network

II.3 Openflow

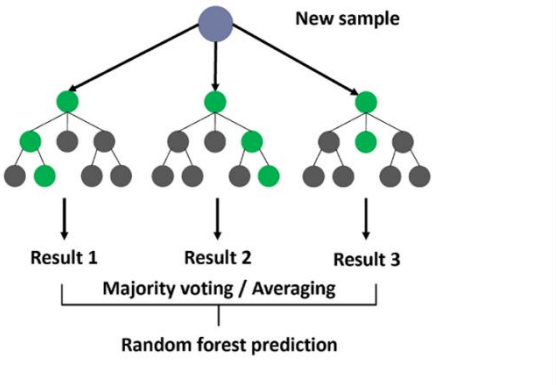
OpenFlow adalah sebuah protokol komunikasi yang digunakan dalam arsitektur Software Defined Networking (SDN). Protokol ini memungkinkan SDN Controller untuk berkomunikasi dengan perangkat keras jaringan, seperti switch dan router, untuk mengontrol aliran lalu lintas dan memanipulasi tabel aliran di perangkat tersebut. OpenFlow menyediakan antarmuka standar yang didefinisikan oleh Open Networking Foundation (ONF) untuk mengatur dan mengelola lalu lintas jaringan.



Gambar II.3 OpenFlow

II.4 Random Forest

Random Forest merupakan kumpulan pohon prediktor di mana setiap pohon bergantung pada nilai vektor acak yang diambil sampelnya secara independen dan didistribusikan secara seragam untuk semua pohon dalam hutan. Kesalahan generalisasi untuk hutan akan konvergen hampir pasti ke suatu batas seiring dengan bertambahnya jumlah pohon dalam hutan. Kesalahan generalisasi dari hutan pengklasifikasi pohon bergantung pada kekuatan masing-masing pohon dalam hutan dan korelasi antar pohon tersebut (Breiman,2001).



Gambar II.4 Random Forest

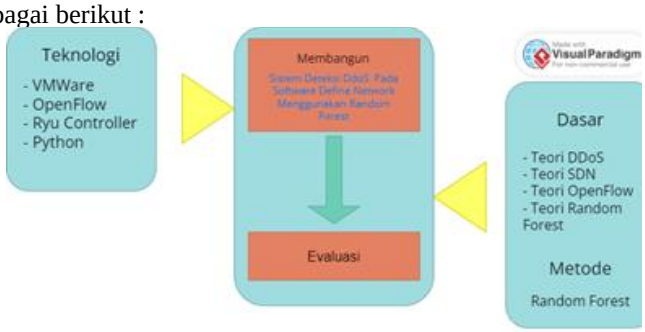
II.5 Ryu Controller

Ryu controller adalah sebuah pengontrol yang digunakan pada SDN dan dijalankan dengan bahasa pemrograman python. Ryu menyediakan komponen software dengan API yang membantu dalam manajemen dan melakukan kontrol pada jaringan SDN, serta memudahkan pengguna dalam melakukan pengembangan aplikasi pada controller. Ryu mensupport berbagai protokol pengelola jaringan seperti Netconf, OpenFlow, dan OF-confg. Ryu terdiri dari 3 lapisan, yang pertama application layer, control layer, dan infrastructure layer. Kegunaan Ryu sendiri adalah untuk melakukan analitik dengan menggunakan algoritma, mengatur atau membuat aturan baru dengan bantuan controller, serta mendapatkan traffic dari controller.

III. METODE PENELITIAN

III.3 Model Konseptual

Model konseptual merupakan representasi visual atau gambaran yang membantu pemahaman, implementasi, dan evaluasi penelitian sistem informasi. Model ini mencakup tujuan, tugas, masalah, serta peluang yang terkait. Fokus utama dari model ini adalah menyediakan kerangka terstruktur yang berguna untuk memahami tujuan dari penelitian tersebut. Dalam konteks penelitian ini, digunakan model konseptual sebagai panduan, yang mencakup elemen-elemen seperti tujuan, tugas, masalah, dan peluang. Model konseptual yang digunakan dalam penelitian ini adalah sebagai berikut :

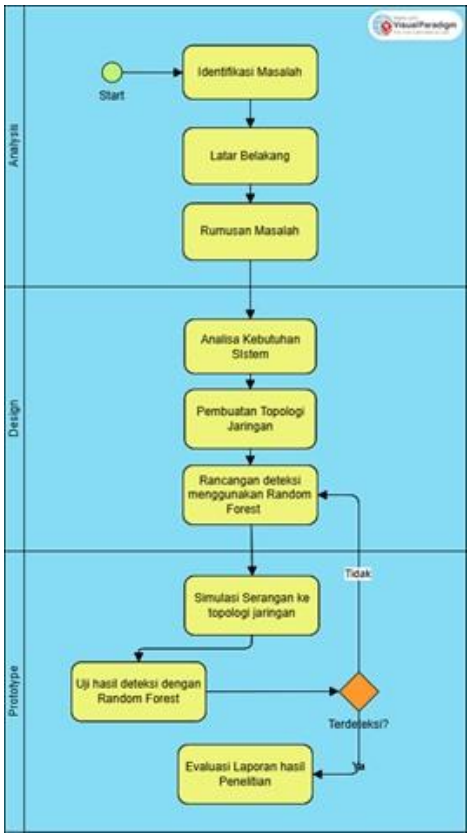


Gambar III.1 Model Konseptual

Dalam gambar III.1, dijelaskan bahwa penelitian mengenai Sistem deteksi dan serangan DDOS pada jaringan SDN menggunakan metode Random Forest memiliki teknologi yang diterapkan meliputi VirtualBox, OpenFlow, dan RYU Controller. Sementara itu, dalam lingkup Penelitian terdapat dua artefak, yaitu artefak "Membangun" yang merincikan aktivitas yang dilakukan selama penelitian, dan "Evaluasi" yang menunjukkan keberhasilan penelitian serta kemampuan untuk menjadi dasar pengembangan lebih lanjut.

Proses evaluasi sistem deteksi serangan DDoS yang menggunakan algoritma Random Forest pada Software Defined Network (SDN) melibatkan beberapa langkah penting untuk memastikan kinerja model. Langkah pertama adalah mengumpulkan data evaluasi yang mencakup traffic normal dan traffic serangan DDoS, baik dari dataset publik maupun hasil simulasi dengan alat seperti hping3, sFlow, atau Wireshark. Selanjutnya, data dibagi menjadi data latih dan data uji, di mana data uji digunakan untuk mengevaluasi kemampuan model dalam mendeteksi serangan secara akurat. Evaluasi dilakukan menggunakan metrik seperti akurasi, precision, recall, F1-score, dan confusion matrix untuk mendapatkan gambaran performa model secara detail. Selain itu, waktu pemrosesan juga dihitung untuk memastikan deteksi dapat dilakukan secara real-time dengan latensi minimum.

III.2 Sistematika Penyelesaian Masalah



Gambar III.2 Sistematika Penyelesaian Masalah

Gambar III-2 menggambarkan suatu sistematisasi penyelesaian masalah, yakni suatu rangkaian langkah atau proses terstruktur yang dijalankan untuk mencapai tujuan akhir dari penelitian. Dimulai dari langkah awal berupa identifikasi masalah yang menjadi fokus dalam penelitian, hingga langkah akhir berupa analisis dan penyimpuan dari hasil penelitian yang telah dilakukan.

III.3 Pengumpulan Data

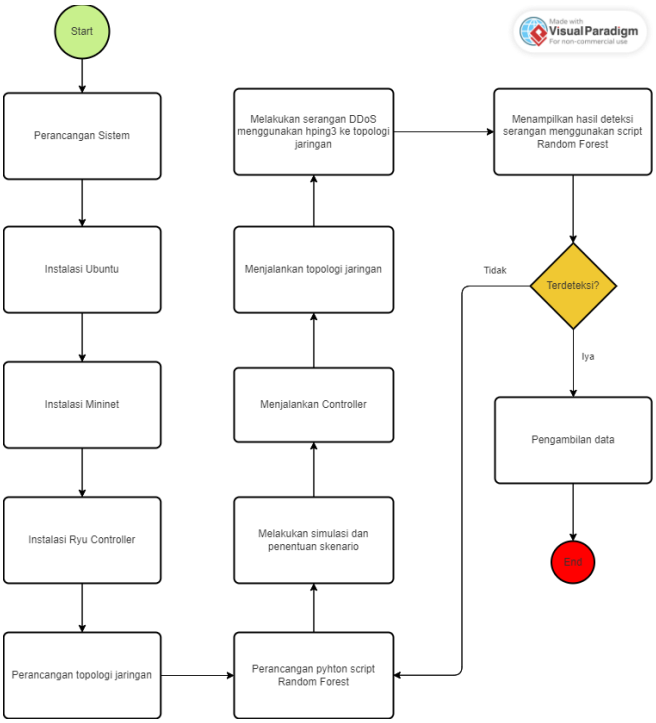
Penelitian ini memerlukan data yang bervariasi, sehingga peneliti menggunakan berbagai teknik pengumpulan data untuk memperoleh informasi yang diperlukan, yaitu :

1. Studi Pustaka, peneliti mengumpulkan berbagai topik sebagai referensi untuk mencapai tujuan penelitian. Referensi tersebut dapat berupa paper, jurnal, buku, dan sumber informasi lainnya.
2. Data, dalam metode ini, penelitian menggunakan dataset yang dihasilkan selama implementasi untuk diuji pada tahap pengujian.
3. Pengujian, setelah mengamati hasil dari penelitian, peneliti segera melakukan pengujian dengan menggunakan perangkat dan alat yang telah ditetapkan, termasuk VMware, RYU Controller, dan OpenFlow.

IV. PERANCANGAN SISTEM DAN PENGUJIAN SKENARIO

IV.1 Alur Perancangan

Berikut merupakan *flowchart* untuk alur perancangan sistem deteksi serangan DDoS menggunakan metode *Random Forest* :



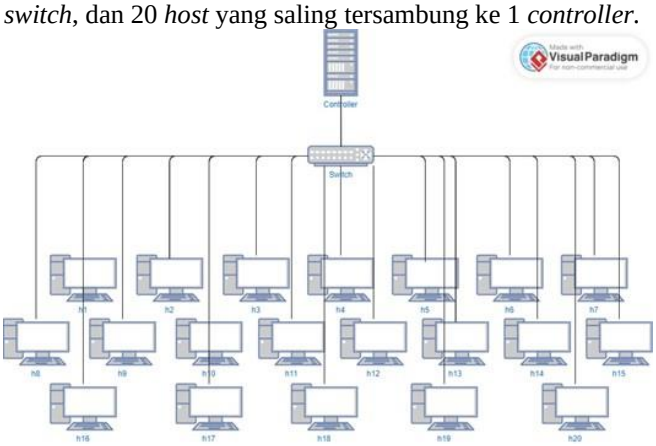
Gambar IV.1 Alur Perancangan

Berikut penjelasan lebih detail mengenai *flowchart* alur perancangan :

1. Melakukan instalasi Ubuntu yang berfungsi sebagai sistem operasi yang digunakan untuk pengujian Tugas Akhir (TA).
2. Dalam sistem operasi Ubuntu, melakukan tahap instalasi Mininet dan controller. Controller yang digunakan yaitu Ryu Controller.
3. Membuat perancangan topologi jaringan SDN pada Mininet. Di dalam topologi akan dirancang jaringan berupa 1 controller, 1 switch, dan 20 host.
4. Membuat rancangan script python untuk model Random Forest dan perhitungan modelnya.
5. Melakukan tes simulasi untuk topologi jaringan dan controller untuk dijalankan secara bersamaan. Menjalankan pingall untuk melakukan pengecekan keberhasilan simulasi tersebut.
6. Menggunakan hping3 untuk melakukan serangan DDoS ke topologi jaringan. Hasil serangan akan dicetak ke file csv.
7. Melakukan deteksi menggunakan script python model Random Forest yang telah dibuat dan mencatat nilai accuracy, precision, recall, dan F1-Score hasil serangan yang telah dicetak ke file csv.

IV.2 Topologi Jaringan

Di subbab ini menunjukkan topologi jaringan yang akan diimplementasikan pada penelitian ini. Topologi ini dibuat dengan menggunakan *tools simulator* Mininet dan berbentuk jaringan SDN. Desain topologi ini terdiri dari 1 *controller*, 1 *switch*, dan 20 *host* yang saling tersambung ke 1 *controller*.



Gambar IV.2 Topologi Jaringan

1. Controller merupakan komponen penting yang berfungsi untuk mengelola dan mengatur seluruh jaringan pada topologi jaringan tersebut. Jenis controller yang digunakan untuk penelitian ini adalah Ryu Controller.
2. Switch merupakan komponen pada topologi jaringan yang berfungsi sebagai jembatan penghubung antara controller dengan host untuk berkomunikasi. Switch berfungsi juga untuk meneruskan paket data berdasarkan perintah yang diterima dari controller.
3. Host merupakan perangkat yang terhubung ke jaringan dan mampu mengirim atau menerima data.

IV.3 Perangkat dan Klasifikasi Host

Pada subbab ini dijelaskan mengenai perangkat yang digunakan dan tabel yang berisikan keterangan IP Address pada host yang akan digunakan dalam penelitian ini menggunakan simulasi tools Mininet.

Tabel IV.1 Perangkat Topologi Jaringan

No	Perangkat	Jumlah
1	Controller	1
2	Switch	1
3	PC (Personal Computer)	20

Tabel IV.2 Klasifikasi Host

No	Host	IP Address	MAC Address
1	H1	10.0.0.1	00:00:00:00:00:01
2	H2	10.0.0.2	00:00:00:00:00:02
3	H3	10.0.0.3	00:00:00:00:00:03
4	H4	10.0.0.4	00:00:00:00:00:04

5	H5	10.0.0.5	00:00:00:00:00:05
6	H6	10.0.0.6	00:00:00:00:00:06
7	H7	10.0.0.7	00:00:00:00:00:07
8	H8	10.0.0.8	00:00:00:00:00:08
9	H9	10.0.0.9	00:00:00:00:00:09
10	H10	10.0.0.10	00:00:00:00:00:010
11	H11	10.0.0.11	00:00:00:00:00:011
12	H12	10.0.0.12	00:00:00:00:00:012
13	H13	10.0.0.13	00:00:00:00:00:013
14	H14	10.0.0.14	00:00:00:00:00:014
15	H15	10.0.0.15	00:00:00:00:00:015
16	H16	10.0.0.16	00:00:00:00:00:016
17	H17	10.0.0.17	00:00:00:00:00:017
18	H18	10.0.0.18	00:00:00:00:00:018
19	H19	10.0.0.19	00:00:00:00:00:019
20	H20	10.0.0.20	00:00:00:00:00:020

IV.4 Akurasi Machine Learning

Machine learning berperan penting dalam meningkatkan efektivitas deteksi serangan secara otomatis. Tujuan utamanya adalah mengenali pola lalu lintas jaringan dan membedakan antara trafik normal dan serangan DDoS berdasarkan karakteristik tertentu, seperti jumlah dan ukuran paket serta interval waktu antar paket. Dengan pendekatan Random Forest, sistem dapat menganalisis data secara efisien, mengidentifikasi anomali dengan akurasi tinggi, serta mengurangi kesalahan dalam klasifikasi serangan. Selain itu, metode ini mampu menangani data dalam jumlah besar dengan lebih cepat dan tetap mempertahankan kinerja yang baik meskipun terdapat fitur yang tidak terlalu relevan. Fungsi utamanya dalam penelitian ini adalah mengotomatisasi proses deteksi, meningkatkan keamanan jaringan SDN, serta membantu administrator dalam mengelola dan menangani serangan DDoS secara lebih efektif.

Rumus akurasi dalam Machine Learning merupakan ukuran untuk menilai seberapa baik model prediktif dapat mengklasifikasikan atau memprediksi label data dengan benar. Akurasi dihitung dengan membandingkan prediksi model terhadap label sebenarnya, dan dinyatakan sebagai persentase dari prediksi yang benar. Berikut rumus akurasi :

$$\boldsymbol{Akurasi = \frac{Jumlah\ Prediksi\ Benar}{Total\ Jumlah\ Prediksi}}$$

Jika dihitung secara sistematis, maka bisa ditulis dengan rumus :

$$\boldsymbol{Akurasi = \frac{TP + TN}{TP + TN + FP + FN}}$$

Keterangan dari rumus diatas yaitu :

1. TP (True Positive) adalah jumlah prediksi positif yang benar.
2. TN (True Netative) adalah jumlah prediksi negatif yang benar.
3. FP (False Positive) adalah julah prediksi positif yang salah.
4. FN (False Negative) adalah jumlah prediksi negatif yang salah.

IV.5 Precision

Precision adalah salah satu metrik evaluasi dalam Machine Learning yang digunakan untuk mengukur seberapa akurat prediksi model dalam mengidentifikasi kelas positif dari seluruh prediksi yang dinyatakan sebagai positif. *Precision* memberikan informasi tentang tingkat kesalahan yang dihasilkan oleh model saat menyatakan suatu sampel sebagai positif. Berikut rumus dari *Precision* :

$$\boldsymbol{Precision = \frac{TP}{TP + FP}}$$

IV.5.1 Recall

Recall adalah metrik evaluasi dalam Machine Learning yang digunakan untuk mengukur seberapa baik model dapat mengidentifikasi semua sampel yang sebenarnya positif. *Recall* memberikan informasi tentang seberapa efektif model dalam menemukan semua kasus positif dari dataset. Berikut rumus *Recall* :

$$\boldsymbol{Recall = \frac{TP}{TP + FN}}$$

IV.5.2 F1-Score

F1 Score adalah metrik evaluasi dalam Machine Learning yang menggabungkan precision dan recall ke dalam satu nilai, yang memberikan gambaran tentang keseimbangan antara keduanya. *F1 Score* sangat berguna ketika Anda ingin mengevaluasi model yang bekerja dengan dataset yang tidak seimbang atau ketika ada kebutuhan untuk mempertimbangkan trade-off antara *precision* dan *recall.* Secara matematis, *F1 Score* adalah rata-rata harmonik dari *Precision* dan *Recall*, yang dinyatakan dengan rumus :

$$\boldsymbol{F1\ Score = 2\ X\ \frac{P\ x\ R}{P + R}}$$

V. ANALISA DAN EVALUASI HASIL PERANCANGAN

V.1 Implementasi Sistem

Pada tahap ini, dilakukan implementasi sistem deteksi serangan DDoS yang telah dirancang. Sistem menggunakan arsitektur Software-Defined Networking (SDN) dengan

topologi jaringan yang terdiri dari 20 host yang terkoneksi melalui satu switch dan dikendalikan oleh Ryu Controller.
Serangan DDoS disimulasikan menggunakan beberapa host (h1, h3, h10, h13) yang mengirimkan lalu lintas berbasis protokol TCP/UDP dengan menggunakan tool hping3 ke target (h20). Sementara itu, lalu lintas normal (ping) dikirimkan oleh host h15 ke h20. Pengumpulan data lalu lintas dilakukan oleh controller yang merekam parameter seperti ukuran paket, tingkat lalu lintas, waktu antar-paket, protokol, dan label. Data ini digunakan untuk melatih dan menguji model Random Forest untuk mendeteksi serangan DDoS.
Untuk mendukung implementasi deteksi serangan DDoS pada jaringan berbasis SDN, topologi jaringan dirancang menggunakan Mininet. Script Python berikut mendefinisikan topologi yang terdiri dari satu switch, 20 host, dan satu controller.

```
from mininet.net import Mininet
from mininet.node import RemoteController
from mininet.cli import CLI
from mininet.log import setLogLevel

def simple_topo():
    net = Mininet(controller=RemoteController)

    print("Creating controller")
    net.addController('c0', controller=RemoteController, ip='127.0.0.1', port=6633)

    print("Creating hosts")
    h1 = net.addHost('h1', mac='00:00:00:00:00:01')
    h2 = net.addHost('h2', mac='00:00:00:00:00:02')
    h3 = net.addHost('h3', mac='00:00:00:00:00:03')
    h4 = net.addHost('h4', mac='00:00:00:00:00:04')
    h5 = net.addHost('h5', mac='00:00:00:00:00:05')
    h6 = net.addHost('h6', mac='00:00:00:00:00:06')
    h7 = net.addHost('h7', mac='00:00:00:00:00:07')
    h8 = net.addHost('h8', mac='00:00:00:00:00:08')
    h9 = net.addHost('h9', mac='00:00:00:00:00:09')
    h10 = net.addHost('h10', mac='00:00:00:00:00:10')
    h11 = net.addHost('h11', mac='00:00:00:00:00:11')
    h12 = net.addHost('h12', mac='00:00:00:00:00:12')
    h13 = net.addHost('h13', mac='00:00:00:00:00:13')
    h14 = net.addHost('h14', mac='00:00:00:00:00:14')
    h15 = net.addHost('h15', mac='00:00:00:00:00:15')
    h16 = net.addHost('h16', mac='00:00:00:00:00:16')
    h17 = net.addHost('h17', mac='00:00:00:00:00:17')
    h18 = net.addHost('h18', mac='00:00:00:00:00:18')
    h19 = net.addHost('h19', mac='00:00:00:00:00:19')
    h20 = net.addHost('h20', mac='00:00:00:00:00:20')


    print("Creating switch")
    s1 = net.addSwitch('s1', protocols='OpenFlow13')

    print("Creating links")
    net.addLink(h1, s1)
    net.addLink(h2, s1)
    net.addLink(h3, s1)
    net.addLink(h4, s1)
    net.addLink(h5, s1)
    net.addLink(h6, s1)
    net.addLink(h7, s1)
    net.addLink(h8, s1)
    net.addLink(h9, s1)
    net.addLink(h10, s1)
    net.addLink(h11, s1)
    net.addLink(h12, s1)
    net.addLink(h13, s1)
    net.addLink(h14, s1)
    net.addLink(h15, s1)
    net.addLink(h16, s1)
    net.addLink(h17, s1)
    net.addLink(h18, s1)
    net.addLink(h19, s1)
```

```
    net.addLink(h20, s1)

    print("Starting network")
    net.start()

    print("Running CLI")
    CLI(net)

    print("Stopping network")
    net.stop()


if __name__ == '__main__':
    setLogLevel('info')
    simple_topo()
```

Jalankan script di atas menggunakan Python untuk membuat jaringan SDN dengan Mininet.

```
sudo python3 topologi.py
```

```
root@ubuntu2004:/home/ubuntu/TA# python3 topologi.py
Creating controller
Creating hosts
Creating switch
Creating links
Starting network
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20
*** Starting controller
c0
*** Starting 1 switches
s1 ...
Running CLI
*** Starting CLI:
mininet>
```

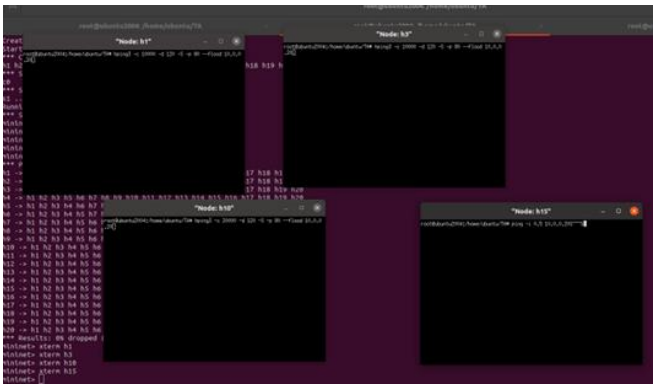
Gambar V.1.1 Menjalankan Mininet

Jalankan script Ryu Controller yang telah disiapkan untuk mengcapture traffic.

```
root@ubuntu2004:/home/ubuntu# cd TA
root@ubuntu2004:/home/ubuntu/TA# ls
Confusion.png  data1.csv                __pycache__   TopoAlam.py
cont1.py       ddos_detector_model.pkl  rafo1.py      topologi.py
Cont.py        fileh1.txt               rafo.py       Topo.py
Conttest.py    generate.py              Roc.png       TopoTest.py
root@ubuntu2004:/home/ubuntu/TA# nano cont1.py
root@ubuntu2004:/home/ubuntu/TA# nano Cont.py
root@ubuntu2004:/home/ubuntu/TA# ryu-manager Cont.py
loading app Cont.py
loading app ryu.controller.ofp_handler
instantiating app Cont.py of EnhancedPacketMonitor
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Gambar V.1.2 Menjalankan Ryu Controller

Untuk menguji lalu lintas normal, gunakan host yang tidak terlibat dalam serangan (contoh: h15) untuk mengirimkan paket ke target (h20). Lalu lakukan serangan oleh beberapa host (h1, h3, h10) yang mengirimkan lalu lintas berbasis TCP atau UDP secara bersamaan ke target (h20) seperti gambar dibawah.



Gambar V.1.3 Traffic Serangan dan Uji Coba Jaringan

Controller akan memproses lalu lintas dan mencatat parameter penting, seperti :

1. IP Source dan Destination
2. Ukuran Paket
3. Tingkat lalu lintas (packet rate)
4. Waktu antar paket (inter-arrival time)
5. Protokol yang digunakan
6. Label serangan atau normal

Data yang dicapture diatas akan diekspor ke file CSV (data1.csv) secara otomatis oleh controller.

V.2 Analisa Hasil Deteksi menggunakan Classification Report

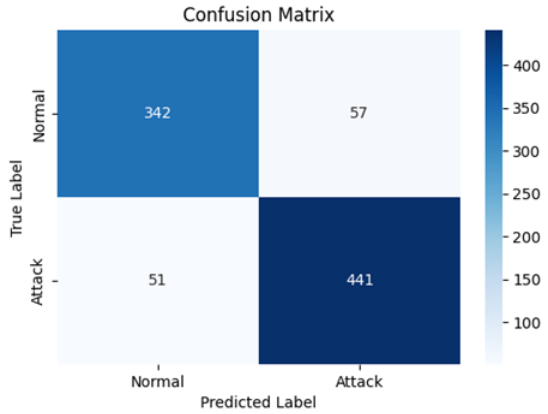
Analisa pertama penulis menggunakan classification report yang menghasilkan nilai *accuracy*, *precision*, *recall* dan *F1-Score* seperti berikut :

```
root@ubuntu2004:/home/ubuntu/TA# python3 rafo1.py
Accuracy: 0.8787878787878788
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.86      0.86       399
           1       0.89      0.90      0.89       492

    accuracy                           0.88       891
   macro avg       0.88      0.88      0.88       891
weighted avg       0.88      0.88      0.88       891
```

Gambar V.2.1 Evaluasi model Random Forest

Dari hasil classification report diatas, dapat disimpulkan bahwa model memiliki kinerja yang cukup baik dalam mengklasifikasikan data. Model ini mampu mengidentifikasi dengan akurasi yang tinggi baik sampel normal maupun sampel serangan DDoS.

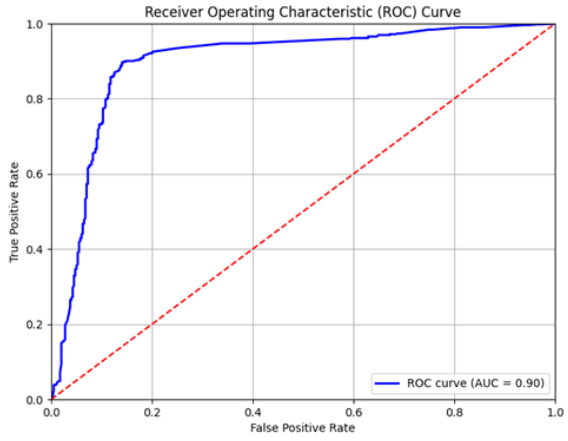
V.3 Confusion Matrix



Gambar V.3.1 Confusion Matrix

Berdasarkan hasil evaluasi model yang ditunjukkan oleh Confusion Matrix, dapat disimpulkan bahwa model deteksi serangan DDoS yang dikembangkan telah menunjukkan kinerja yang cukup baik. Model ini mampu mengklasifikasikan sebagian besar sampel data dengan akurat, baik itu data normal maupun data yang mengandung serangan. Tingkat akurasi yang tinggi dalam mengidentifikasi serangan menunjukkan bahwa model efektif dalam mendeteksi ancaman siber jenis ini. Meskipun demikian, masih terdapat beberapa kasus di mana model melakukan kesalahan klasifikasi, baik dengan mengklasifikasikan data normal sebagai serangan (false positive) maupun sebaliknya (false negative). Hal ini mengindikasikan bahwa terdapat ruang untuk perbaikan lebih lanjut, seperti dengan melakukan fine-tuning pada hyperparameter model, mencoba algoritma yang berbeda, atau menambahkan fitur-fitur baru yang lebih relevan. Secara keseluruhan, hasil evaluasi ini memberikan dasar yang kuat untuk pengembangan sistem deteksi serangan DDoS yang lebih robust dan handal di masa mendatang.

V.4 ROC Curve dan AUC



Gambar V.4.1 ROC Curve

Kurva ROC di atas menunjukkan bahwa model yang digunakan memiliki kinerja yang sangat baik dalam mengklasifikasikan data. Nilai AUC yang tinggi (0.90) mengindikasikan bahwa model mampu membedakan dengan baik antara kelas positif dan negatif. Model ini memiliki kemampuan yang sangat baik dalam mengidentifikasi sampel positif (serangan) tanpa terlalu banyak menghasilkan false positive (mengklasifikasikan sampel negatif sebagai positif).

VI. KESIMPULAN DAN SARAN

VI.1 Kesimpulan

1. Penelitian ini telah berhasil mengembangkan sebuah sistem deteksi serangan DDoS yang efektif dengan menggabungkan kekuatan Software-Defined Networks (SDN) dan algoritma Random Forest.
2. Hasil pengujian menunjukkan bahwa sistem ini mampu mengidentifikasi pola lalu lintas yang mencurigakan dengan akurasi mencapai 88%, sehingga dapat mencegah serangan DDoS sebelum menyebabkan kerusakan yang signifikan. Fleksibilitas SDN memungkinkan sistem ini untuk beradaptasi dengan berbagai jenis serangan DDoS, sementara algoritma Random Forest memberikan kemampuan klasifikasi yang kuat

.

VI.2 Saran

Berdasarkan hasil penelitian ini, beberapa saran dapat diajukan untuk pengembangan lebih lanjut seperti, Pengembangan Fitu untuk Mengidentifikasi fitur-fitur baru yang lebih relevan dengan jenis serangan DDoS terkini dapat meningkatkan akurasi deteksi. Selain itu, Optimasi Algoritma dengan algoritma pembelajaran mesin lainnya, seperti Support Vector Machine (SVM) atau Neural Network, untuk membandingkan performanya dengan Random Forest.

REFERENSI

Gupta, R., & Kumar, M. (2020). "DDoS Attack Detection Using Machine Learning Algorithms in SDN." International Journal of Engineering and Advanced Technology (IJEAT), Vol. 9, Issue 4. DOI: 10.35940/ijeat.D1748.049420.

Azad, M. A., et al. (2020). "Real-Time Detection of DDoS Attacks in SDN Using Machine Learning." Journal of Network and Computer Applications, 169, 102779. DOI: 10.1016/j.jnca.2020.102779.

Kumar, N., et al. (2020). "A Hybrid Approach for Detecting DDoS in SDN Using Random Forest and K-Means Clustering." Wireless Personal Communications, 112, 2741-2760. DOI: 10.1007/s11277-020-07252-3.

Huang, C. Y., & Lin, J. Y. (2021). "DDoS Attack Mitigation in SDN Environment Using Random Forest Classifier." *International Journal of Computing and Digital Systems (IJCDS)*, 10(5). Penelitian ini mengevaluasi algoritma Random Forest dalam mendeteksi dan memitigasi serangan DDoS pada lingkungan SDN.

Zhang, J., Chen, Y., & Xiang, Y. (2013). "An Adaptive Method for Detecting DDoS Attacks in SDN." *IEEE Transactions on Computers*, 62(11), 2273-2285.

Yoon, J., Park, S., & Chung, T. (2020). "DDoS Attack Detection Mechanism Using Feature Selection and Ensemble

Learning on Software-Defined Networks." *Journal of Network and Computer Applications*, 164, 102682.

Moustafa, N., & Slay, J. (2016). *"The significant features of the UNSW-NB15 and the KDD99 datasets for network intrusion detection: Statistical analysis and recommendations."* Pada penelitian ini, data digunakan untuk membandingkan kinerja berbagai metode, termasuk Random Forest, untuk deteksi intrusi jaringan. DOI: 10.1109/TrustCom.2016.0139

Deka, G., & Bezboruah, A. (2020). *"A hybrid machine learning approach for DDoS detection using Random Forest and PSO."* Studi ini menunjukkan bagaimana penggabungan metode Random Forest dengan optimasi dapat meningkatkan akurasi deteksi DDoS.

Ashfaq, R. A. R., et al. (2017). *"Fuzziness based semi-supervised learning approach for intrusion detection system."* Artikel ini membahas peran metode Random Forest dalam mendeteksi berbagai jenis serangan, termasuk DDoS. DOI: 10.1016/j.ins.2017.03.027

.