 ISSN:2302-8149
e-ISSN:2540-9719

Penerapan Sistem Keamanan Jaringan menggunakan *Intrusion Prevention System* dengan *Machine Learning*

Implementation of a Network Security System using an Intrusion Prevention System with Machine Learning

[1]Andre Pardamean Lumban Gaol*, [2]Agus Tedyyana, [3]Nurmi Hidayasari
[1,2,3]Program Studi Keamanan Sistem Informasi, Jurusan Teknik Informatika, Politeknik Negeri Bengkalis,
[1,2,3]Jl. Bathin Alam, Sungai Alam, Bengkalis, Indonesia
*e-mail: andrepardamean03@gmail.com

(***received*:** 24 June 2025, ***revised*:** 26 July 2025, ***accepted*:** 29 July 2025)

Abstrak

Penelitian ini melakukan pengembangan *Intrusion Prevention System (IPS)* berbasis machine learning untuk mendeteksi dan mencegah serangan jaringan secara otomatis. Sistem dirancang menggunakan algoritma *Random Forest* yang dilatih dengan data dari CICIDS2017 dan CICIDS2019, yaitu dataset standar yang dikembangkan oleh *Canadian Institute for Cybersecurity* dan banyak digunakan dalam penelitian keamanan siber karena menyediakan data lalu lintas jaringan realistis dengan berbagai jenis serangan. Sistem difokuskan pada tiga jenis serangan umum: *Syn Flood*, *Port Scanning*, dan *SSH Patator*. Setelah melalui proses pra-pemrosesan, pelatihan, dan evaluasi, model diintegrasikan ke dalam sistem IPS yang mampu melakukan pemantauan jaringan secara real-time, memblokir IP penyerang, dan mengirimkan notifikasi otomatis melalui Telegram. Hasil pengujian menunjukkan bahwa sistem mampu mendeteksi serangan dengan akurasi tinggi serta memberikan respons cepat dan efisien. Sistem ini memberikan kemudahan bagi administrator jaringan dalam mendeteksi dan merespons serangan tanpa harus melakukan pemantauan log secara manual. Dengan pendekatan otomatis dan adaptif, sistem IPS ini berkontribusi nyata dalam peningkatan keamanan jaringan dan dapat diimplementasikan langsung pada lingkungan jaringan organisasi atau institusi untuk mengurangi risiko serangan siber secara signifikan.

Kata kunci: *Intrusion Prevention System, Machine Learning, Random Forest*, Keamanan Jaringan, Notifikasi Telegram

Abstract

This research develops a machine learning-based Intrusion Prevention System (IPS) to automatically detect and prevent network attacks. The system was designed using the Random Forest algorithm, trained on the CICIDS2017 and CICIDS2019 datasets—standard benchmarks developed by the Canadian Institute for Cybersecurity, widely used in cybersecurity research for their realistic network traffic and diverse attack types. The system focuses on three common attacks: SYN Flood, Port Scanning, and SSH Patator. After preprocessing, training, and evaluation, the model was integrated into the IPS, enabling real-time network monitoring, attacker IP blocking, and automated notifications via Telegram. Testing results indicate that the system achieves high detection accuracy while delivering fast and efficient responses. This system simplifies the work of network administrators by detecting and responding to attacks without the need for manual log monitoring. Through its automated and adaptive approach, the IPS makes a significant contribution to enhancing network security and can be directly implemented in organizational or institutional network environments to substantially reduce the risk of cyberattacks.

Keywords: *Intrusion Prevention System, Machine Learning, Random Forest, Network Security, Telegram Notification*

 ISSN:2302-8149
e-ISSN:2540-9719

1 Pendahuluan

Kemajuan pada ilmu pengetahuan dan teknologi seperti saat ini menjadikan infrastruktur jaringan komputer menjadi bagian yang penting dalam berbagai kegiatan bisnis, pendidikan, pemerintahan, dan layanan publik lainnya[1]. Jaringan komputer dimanfaatkan untuk pertukaran data yang cepat dan aman dalam mendukung operasional sistem. Namun, tingginya ketergantungan terhadap jaringan juga meningkatkan risiko serangan siber, seperti pencurian data dan gangguan layanan. Penerapan kebijakan keamanan, teknologi proteksi, dan pelatihan berkala menjadi penting dalam mitigasi risiko[2]. Jumlah dan kompleksitas serangan siber terus mengalami peningkatan, khususnya terhadap jaringan yang terhubung ke *internet*. Beberapa jenis serangan yang umum terjadi meliputi *SYN flood*, *port scanning*, dan *SSH Patator*, yang dapat mengakibatkan gangguan layanan, pencurian atau kerusakan data, serta potensi kehilangan data penting[3]. Ketiga serangan tersebut dipilih sebagai fokus penelitian karena mewakili ancaman nyata yang sering terjadi di lingkungan jaringan *modern*. *SYN Flood* dapat melumpuhkan layanan dengan membanjiri permintaan koneksi, *Port Scanning* menjadi pintu awal eksploitasi celah keamanan, sedangkan *SSH Patator* mengincar kredensial *login* secara *brute force*. Berbagai sistem *Intrusion Prevention System* (*IPS*) konvensional masih memiliki keterbatasan, seperti bergantung pada pola serangan yang telah diketahui (*signature-based*), tidak adaptif terhadap pola baru, serta minim otomatisasi dalam merespons ancaman. Oleh karena itu, diperlukan pendekatan baru yang lebih adaptif dan efisien. Salah satu solusi yang ditawarkan dalam penelitian ini adalah pengembangan sistem IPS berbasis machine learning, yang mampu mengenali pola serangan secara adaptif dan memberikan respons secara real-time. Teknologi ini dinilai efektif dalam meningkatkan ketahanan dan keamanan jaringan dari berbagai ancaman siber[4].

Keamanan jaringan merupakan elemen krusial dalam melindungi sistem dari aktivitas ilegal yang dapat mengakibatkan pencurian data maupun kerusakan sistem. Keamanan ini mencakup serangkaian tindakan dan teknologi yang dirancang untuk menjaga integritas, kerahasiaan, dan ketersediaan data serta sumber daya dalam jaringan. Upaya tersebut bertujuan untuk melindungi infrastruktur jaringan dari berbagai ancaman, termasuk serangan siber, akses tidak sah, pencurian data, kerusakan perangkat, maupun kesalahan manusia (*human error*)[5]. Untuk mencapai tingkat perlindungan yang optimal, diperlukan pendekatan keamanan berlapis. Salah satu strategi yang dapat diimplementasikan adalah penerapan *Intrusion Prevention System* (*IPS*), yaitu teknologi keamanan yang dirancang untuk mendeteksi dan mencegah serangan secara otomatis. *IPS* bekerja dengan memantau lalu lintas jaringan secara *real-time* dan mengambil tindakan terhadap aktivitas yang mencurigakan atau berpotensi mengganggu kinerja jaringan[6]. *Machine Learning* (*ML*) merupakan cabang dari *Artificial Intelligence* (*AI*) yang memungkinkan sistem untuk belajar dari data dan menghasilkan prediksi atau keputusan tanpa diprogram secara eksplisit. Secara umum, *ML* menggunakan pendekatan komputasi statistik untuk membangun *model* matematika dari data pelatihan, sehingga mampu melakukan prediksi berbasis pola[7]. Salah satu algoritma yang banyak digunakan dalam *ML* adalah *Random Forest*, yang termasuk dalam metode *ensemble learning*. Algoritma ini membangun sejumlah *decision tree* dari *subset* acak data pelatihan dan fitur, kemudian menggabungkan hasil prediksi dari seluruh pohon untuk menghasilkan keputusan akhir. Pendekatan ini tidak hanya meningkatkan akurasi dan stabilitas prediksi, tetapi juga mampu mengurangi risiko *overfitting*. Selain itu, *Random Forest* memiliki kemampuan untuk mengevaluasi pentingnya fitur dalam proses prediksi, sehingga bermanfaat dalam analisis hubungan antar variabel[8].

Penelitian ini bertujuan untuk membangun sistem *IPS* yang mampu mendeteksi dan mencegah serangan *Syn Flood, Port Scanning,* dan *SSH Patator* secara otomatis menggunakan *model* yang telah dilatih dengan algoritma *Machine Learning*. Selain itu, sistem ini dirancang untuk memberikan respons cepat dengan memblokir alamat IP penyerang serta mengirimkan notifikasi pemblokiran secara otomatis melalui *Telegram*. Adapun manfaat dari pengembangan sistem ini adalah untuk mengurangi risiko ancaman keamanan pada jaringan melalui deteksi dan pencegahan serangan secara *real-time*, serta memberikan kemudahan bagi *administrator* jaringan dalam memantau insiden keamanan tanpa perlu melakukan pengawasan *manual* secara terus-menerus.

2 Tinjauan Literatur

Berbagai penelitian telah mengkaji pengembangan *Intrusion Prevention System* (*IPS*) sebagai upaya meningkatkan keamanan jaringan dari beragam ancaman siber. Penelitian sebelumnya

 ISSN:2302-8149
e-ISSN:2540-9719

menunjukkan efektivitas penggunaan *Snort* dan *IPTables* dalam mendeteksi serta mencegah serangan *brute force* dan *port scanning*. Namun, kemampuan *Snort* dalam menghadapi serangan *DDoS* masih terbatas, karena pola serangan *HTTP* lambat yang tidak mudah dikenali oleh aturan yang ada. Solusi yang disarankan dalam studi tersebut adalah penggunaan deteksi berbasis anomali dan integrasi teknologi seperti *SDN* dan *load balancer* untuk meningkatkan efektivitas sistem[9]. Studi lain menambahkan komponen *Honeypot* ke dalam sistem *IPS* berbasis *Snort* dalam lingkungan *SDN*, yang berhasil meningkatkan *throughput* dan menurunkan *packet loss*. Meskipun demikian, keberhasilan sistem sangat tergantung pada kualitas *rule Snort*, dan generalisasi hasil ke berbagai kondisi jaringan masih menjadi kendala[10]. Seiring berkembangnya pendekatan *machine learning* dalam keamanan jaringan, berbagai penelitian mulai memanfaatkan algoritma pembelajaran mesin untuk meningkatkan akurasi dan adaptivitas dalam mendeteksi serangan. Salah satunya adalah kombinasi *Principal Component Analysis* (*PCA*) dan *Random Forest* yang terbukti mampu meningkatkan akurasi deteksi serangan hingga 96,78%. Namun, pengujian terbatas pada skenario simulasi membuat validitas untuk lingkungan nyata masih diragukan[11]. Penelitian lain memperkenalkan *Crowdsec* sebagai *IDPS* yang diimplementasikan di seluruh layanan infrastruktur *server* dan terhubung dengan *Local API*. Hasilnya, sistem mampu secara otomatis memblokir akses IP berbahaya secara otomatis dan terdistribusi, tetapi belum membahas integrasi pembelajaran mesin atau fokus pada jenis serangan tertentu[12]. Sementara itu, penelitian lainnya menggunakan metode *traffic behavior* dengan *Suricata* dan *pfSense* untuk mendeteksi serta memblokir serangan *DDoS*, *brute force*, dan *port scanning*. Pendekatan ini menunjukkan efektivitas tinggi dalam memberikan perlindungan jaringan secara *real-time*, meskipun belum membahas otomasi notifikasi atau integrasi dengan sistem komunikasi *real-time*[13]. Di sisi lain, penelitian menggunakan *Fail2ban* yang dikombinasikan dengan *Python* untuk membangun *dashboard monitoring* berbasis *web* guna menangani serangan *SSH brute force*. Penelitian ini menyoroti pentingnya kemudahan pemantauan bagi *administrator* sistem dan efektivitas *Fail2ban* dalam memblokir ribuan IP penyerang secara otomatis[14]. Penelitian lain berfokus pada simulasi deteksi dan pencegahan serangan *DDoS* pada jaringan berbasis *Software Defined Network* (*SDN*), dengan menggunakan kombinasi *Intrusion Detection System* (*IDS*) dan *Intrusion Prevention System* (*IPS*). Studi ini menguji efektivitas dua pendekatan deteksi, yaitu berbasis signature dan anomaly. Hasilnya menunjukkan bahwa *Signature-IDS* mampu mendeteksi serangan seperti *ICMP Flood*, *SYN Flood*, dan *UDP Flood* dengan waktu respons yang lebih cepat dibandingkan pendekatan *anomaly* namun belum mengintegrasikan komponen notifikasi ataupun sistem pembelajaran mesin secara menyeluruh.[15].

Dalam konteks algoritma *machine learning*, *Random Forest* dikenal unggul karena kemampuannya dalam menangani data berdimensi besar, toleransi tinggi terhadap *noise*, dan kemampuan menghasilkan prediksi yang stabil. Algoritma ini bekerja dengan membentuk banyak pohon keputusan (*decision tree*) dari *subset data* acak dan menggabungkan hasilnya melalui proses *voting*, sehingga mampu meningkatkan akurasi sekaligus mengurangi risiko *overfitting*. Selain itu, *Random Forest* juga memiliki kemampuan mengevaluasi pentingnya fitur dalam prediksi, menjadikannya sangat cocok untuk menganalisis pola serangan yang kompleks pada lalu lintas jaringan.

Berdasarkan tinjauan terhadap berbagai penelitian sebelumnya, sebagian besar sistem *IPS* yang dikembangkan menggunakan pendekatan konvensional atau *machine learning*, namun umumnya hanya berfokus pada satu atau dua jenis serangan seperti *brute force*, *DDoS*, atau *port scanning*. Hingga saat ini, belum ditemukan penelitian yang secara khusus mengintegrasikan deteksi tiga jenis serangan *Syn Flood*, *Port Scanning*, dan *SSH Patator* dalam satu sistem *IPS* berbasis *Random Forest*, sekaligus dilengkapi dengan otomatisasi pemblokiran IP dan notifikasi *real-time* melalui *platform* komunikasi seperti *Telegram*. Mayoritas penelitian hanya membahas aspek deteksi atau pencegahan sebagian dari serangan tersebut, tanpa menyertakan mekanisme respons otomatis yang dapat secara langsung membantu administrator dalam penanganan insiden keamanan secara efisien.

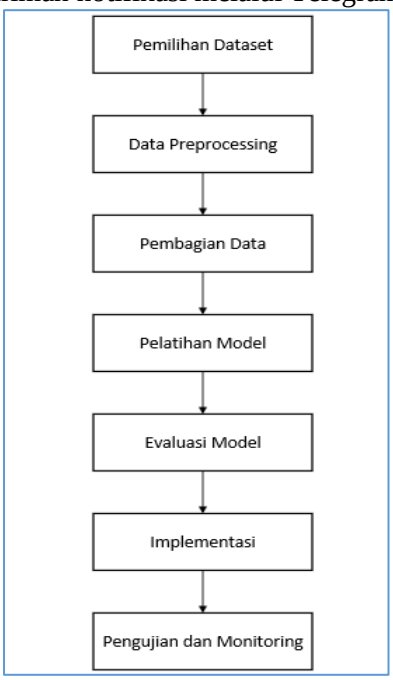
.

3 Metode Penelitian

Penelitian ini menggunakan pendekatan pengembangan sistem berbasis *machine learning* yang terdiri dari beberapa tahapan utama seperti ditunjukkan pada Gambar 1. Tahapan diawali dengan pemilihan *dataset* dari CICIDS2017 dan CICIDS2019, yang merupakan *dataset benchmark* dalam

 ISSN:2302-8149
e-ISSN:2540-9719

penelitian keamanan jaringan. *Dataset* ini dipilih karena menyediakan data lalu lintas jaringan aktual dengan *label* serangan yang jelas dan mencakup jenis serangan yang menjadi fokus penelitian, yaitu *Syn Flood*, *Port Scanning*, dan *SSH Patator*. Selanjutnya dilakukan proses *preprocessing data*, yang mencakup pembersihan data, penyeimbangan jumlah data per kelas serangan agar tidak timpang, serta normalisasi agar seluruh fitur memiliki skala yang seragam. Langkah ini penting agar model tidak bias dan mampu mengenali pola serangan secara konsisten. Data yang telah diproses kemudian dibagi menjadi dua bagian, yaitu data pelatihan dan data pengujian, dengan tujuan agar model dapat dilatih dan diuji secara objektif untuk menghindari *overfitting*. Model deteksi serangan dibangun menggunakan algoritma *Random Forest*, yang dipilih karena mampu menangani data berdimensi tinggi, memberikan akurasi tinggi, serta tahan terhadap *noise*. Evaluasi dilakukan dengan mengukur kinerja *model* berdasarkan metrik seperti akurasi, presisi, dan recall. Setelah model menunjukkan performa yang baik, langkah selanjutnya adalah mengintegrasikan model ke dalam sistem *Intrusion Prevention System* (*IPS*). Pada tahap akhir, sistem diuji dalam lingkungan jaringan dengan mensimulasikan serangan nyata untuk memastikan kemampuan deteksi, pemblokiran otomatis terhadap IP penyerang, serta pengiriman notifikasi melalui Telegram secara real-time.



Gambar 1 Proses pembuatan IPS

3.1 Pengumpulan Dataset

Dataset yang digunakan dalam penelitian ini berasal dari dua sumber terbuka, yaitu CICIDS2017 dan CICIDS2019 yang dikembangkan oleh Canadian Institute for Cybersecurity. Dataset ini mengandung berbagai jenis serangan jaringan, termasuk tiga serangan utama yang menjadi fokus penelitian. Data diunduh dalam format CSV dan berisi berbagai fitur lalu lintas jaringan seperti protokol, durasi koneksi, flag, byte, dan label serangan. Sebelum menggabungkan *dataset* dan melakukan tahapan-tahapan untuk mendapatkan *model* yang akan digunakan untuk IPS, langkah awal yang dilakukan yaitu seperti berikut:

```
import pandas as pd
import numpy as np
import joblib
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from imblearn.under_sampling import RandomUnderSampler
import matplotlib.pyplot as plt
```

 ISSN:2302-8149
e-ISSN:2540-9719

```
import seaborn as sns
from collections import Counter
```

Setelah selesai mengimport library dan pustaka yang dibutuhkan untuk melatih model, selanjutnya peneliti menggabungkan ke tiga dataset agar menjadi 1 *dataset* seperti Gambar 2 berikut ini:

```
# Memuat ketiga dataset
syn_dataset = pd.read_csv(r'D:\tes\syn.csv')
portscan_dataset = pd.read_csv(r'D:\tes\portscan.csv')
bruteforce_dataset = pd.read_csv(r'D:\tes\bruteforce.csv')
```

Gambar 2 Memuat tiga dataset

```
# Menggabungkan ketiga dataset
dataset = pd.concat([syn_dataset, portscan_dataset, bruteforce_dataset], ignore_index=True)
print(f"Dataset berhasil digabungkan, jumlah total data setelah digabungkan: {len(dataset)}")

# Menyimpan dataset yang telah digabungkan ke file CSV
output_file = 'gabungan_dataset.csv'
dataset.to_csv(output_file, index=False)

print(f"Dataset yang telah digabungkan disimpan di file: {output_file}")
```

```
Dataset berhasil digabungkan, jumlah total data setelah digabungkan: 2307119
Dataset yang telah digabungkan disimpan di file: gabungan_dataset.csv
```

Gambar 3 Menggabungkan dan menyimpan dataset

Gambar 3, menunjukkan hasil penggabungan tiga dataset serangan: Syn Flood, Portscan, dan SSH Patator menggunakan fungsi pd.concat() dari pustaka pandas dengan parameter ignore_index=True untuk mereset indeks. Dataset gabungan disimpan ke dalam variabel dataset dan memiliki total 2.307.119 entri. Selanjutnya, dataset disimpan ke file CSV bernama gabungan_dataset.csv menggunakan to_csv(index=False) agar indeks tidak disertakan. Setelah berhasil disimpan, sistem menampilkan lokasi file CSV.

```
# Memuat dataset
dataset = pd.read_csv(r'C:\Users\LENOVO GAMING\gabungan_dataset.csv')

dataset.head()
```

	Flow ID	Source IP	Source Port	Destination IP	Destination Port	Protocol	Timestamp	Flow Duration	Pac
0	172.16.0.5-192.168.50.1-53058-53058-6	172.16.0.5	53058.0	192.168.50.1	53058	6.0	2018-12-01 13:30:30.741451	115799309	
1	172.16.0.5-192.168.50.1-32237-32237-6	172.16.0.5	32237.0	192.168.50.1	32237	6.0	2018-12-01 13:30:30.741452	113973933	
2	172.16.0.5-192.168.50.1-60495-9840-6	172.16.0.5	60495.0	192.168.50.1	9840	6.0	2018-12-01 13:30:30.741501	112	
3	172.16.0.5-192.168.50.1-59724-59724-6	172.16.0.5	59724.0	192.168.50.1	59724	6.0	2018-12-01 13:30:30.741563	105985004	
4	172.16.0.5-192.168.50.1-60496-32538-6	172.16.0.5	60496.0	192.168.50.1	32538	6.0	2018-12-01 13:30:30.741565	1	

5 rows × 87 columns

Gambar 4 Memuat dataset yang sudah disimpan

Gambar 4 menunjukkan proses pemuatan dan peninjauan isi dataset yang telah disimpan untuk memastikan data terbaca dengan baik sebelum tahap pengolahan lebih lanjut. Dataset gabungan yang digunakan terdiri dari 87 fitur, di antaranya flow ID, source IP, source port, destination IP, destination port, protocol, dan timestamp. Setelah dataset berhasil dimuat, tahapan dilanjutkan dengan proses preprocessing data.

 ISSN:2302-8149
e-ISSN:2540-9719

```
# Mengecek label pada Dataset
dataset.Label.value_counts()
```

```
Label
Syn             1582289
BENIGN           560003
PortScan         158930
SSH-Patator        5897
Name: count, dtype: int64
```

Gambar 5 Mengecek label pada dataset setelah digabungkan

Gambar 5, menunjukkan distribusi label dari hasil penggabungan ketiga dataset sebelum dilakukan tahapan pra-pemrosesan data. Dataset gabungan ini memiliki total 2.307.119 entri, yang masing-masing telah dilabeli sesuai dengan jenis lalu lintas jaringan atau serangan yang diwakilinya. Adapun distribusi data untuk setiap label adalah sebagai berikut: kelas *Syn* memiliki jumlah data paling dominan, yaitu sebanyak 1.582.289 entri, kelas *BENIGN* (lalu lintas normal) sebanyak 560.003 entri, *PortScan* sebanyak 158.930 entri, dan *SSH-Patator* dengan jumlah paling sedikit, yaitu 5.897 entri.

3.2 Pra-pemrosesan Data

Pra-pemrosesan dilakukan untuk memastikan kualitas data sebelum digunakan dalam pelatihan model. Tahapan pra-pemrosesan meliputi:

1. Seleksi Fitur: Berdasarkan Gambar 6 dan Gambar 7, menghapus fitur yang tidak relevan atau duplikat guna mengurangi kompleksitas data dan memfokuskan model hanya pada informasi yang signifikan.

```
# Mengecek data duplikat
print("Jumlah data duplikat:", dataset.duplicated().sum())
```

```
Jumlah data duplikat: 97834
```

Gambar 6 Jumlah fitur sebelum dihapus

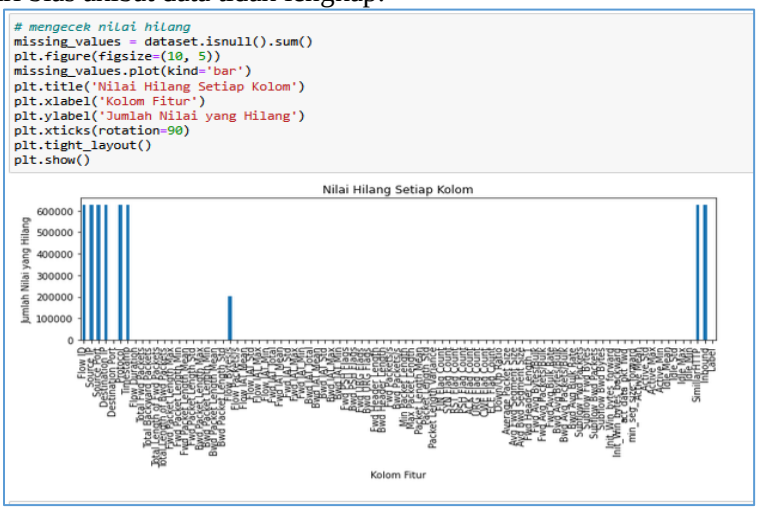
```
# Menghapus data d
dataset = dataset.
print("Jumlah data
```

```
Jumlah data duplik
```

Gambar 7 Jumlah fitur yang telah dihapus

2. Penanganan Nilai Kosong: Berdasarkan Gambar 8 dan Gambar 9, Mengganti nilai *null* dengan rata-rata atau menghapus baris agar tidak mengganggu proses pelatihan dan menghindari bias akibat data tidak lengkap.

```
# mengecek nilai hilang
missing_values = dataset.isnull().sum()
plt.figure(figsize=(10, 5))
missing_values.plot(kind='bar')
plt.title('Nilai Hilang Setiap Kolom')
plt.xlabel('Kolom Fitur')
plt.ylabel('Jumlah Nilai yang Hilang')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

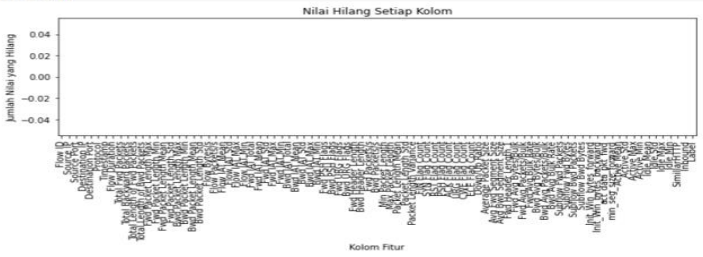


Gambar 8 Cek nilai yang kosong dari setiap kolom

 ISSN:2302-8149
e-ISSN:2540-9719

```
# Mengisi nilai kosong dengan 0
dataset.replace([np.inf, -np.inf], np.nan, inplace=True)
dataset.fillna(0, inplace=True)


# mengecek nilai hilang setelah diatasi
missing_values = dataset.isnull().sum()
plt.figure(figsize=(10, 5))
missing_values.plot(kind='bar')
plt.title('Nilai Hilang Setiap Kolom')
plt.xlabel('Kolom Fitur')
plt.ylabel('Jumlah Nilai yang Hilang')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



Gambar 9 Cek nilai yang hilang dari setiap kolom setelah di hapus

3. Mengecek Jumlah Label: Gambar 10, mengecek jumlah *label* pada *dataset* setelah dilakukan penghapusan data duplikat. Adapun distribusi data berdasarkan label setelah penghapusan data duplikat adalah sebagai berikut: *Syn* sebanyak 1.582.289 entri, *BENIGN* sebanyak 532.958 entri, *PortScan* sebanyak 90.819 entri, dan *SSH-Patator* sebanyak 3.219 entri.

```
# label Dataset Setelah Data Duplikat dihapus
dataset.Label.value_counts()
```

```
Label
Syn             1582289
BENIGN           532958
PortScan          90819
SSH-Patator        3219
Name: count, dtype: int64
```

Gambar 10 Mengecek jumlah label

4. Label Encoding: Gambar 11, mengubah label kategorikal menjadi numerik agar data dapat diproses oleh algoritma machine learning yang hanya mengenali nilai numerik. Pada tahapan ini, dataset memilki 4 label yang mana terdiri dari label benign (lalu lintas normal), syn, portscan dan ssh-patator. Setelah dilakukan encoding, masing-masing label dikonversi menjadi nilai numerik sebagai berikut: label syn akan berubah menjadi 3, label benign (lalu lintas normal) akan berubah menjadi 0, label portscan akan berubah menjadi 1 dan label ssh-patator akan berubah menjadi 2.

```
# Encoding Fitur 'Label' menggunakan LabelEncoder
label_encoder = LabelEncoder()
dataset['Label'] = label_encoder.fit_transform(dataset['Label'])
dataset['Label'].value_counts()
```

```
Label
3    1582289
0     532958
1      90819
2       3219
Name: count, dtype: int64
```

Gambar 11 Encoding fitur dengan label encoder

 ISSN:2302-8149
e-ISSN:2540-9719

5. Normalisasi Data: Gambar 12, menggunakan metode Min-Max Scaling agar semua fitur berada dalam rentang skala yang seragam, sehingga mempercepat konvergensi dan menghindari dominasi fitur tertentu.

```
# Menggabungkan fitur yang sudah dipilih berdasarkan korelasi dengan fitur yang ditambahkan
fitur_baru = fitur + ['RST Flag Count', 'Unique Destination Ports', 'SYN/ACK Ratio', 'Attack Duration']

print("Fitur yang digunakan dalam dataset akhir:", fitur_baru)
```

```
Fitur yang digunakan dalam dataset akhir: ['Source Port', 'Destination Port', 'Protocol', 'Fwd Packet
Length Min', 'Fwd Packet Length Mean', 'Bwd Packet Length Min', 'Bwd Packet Length Mean', 'Bwd Packet
Length Std', 'Flow Packets/s', 'Bwd IAT Total', 'Bwd IAT Mean', 'Bwd IAT Max', 'Fwd PSH Flags', 'Fwd
Packets/s', 'Bwd Packets/s', 'Min Packet Length', 'Max Packet Length', 'Packet Length Mean', 'Packet
Length Std', 'Packet Length Variance', 'SYN Flag Count', 'PSH Flag Count', 'ACK Flag Count', 'URG Fla
g Count', 'Down/Up Ratio', 'Average Packet Size', 'Avg Fwd Segment Size', 'Avg Bwd Segment Size', 'In
it_Win_bytes_backward', 'min_seg_size_forward', 'Active Mean', 'Active Max', 'Idle Mean', 'Idle Min',
'Inbound', 'RST Flag Count', 'Unique Destination Ports', 'SYN/ACK Ratio', 'Attack Duration']
```

Gambar 12 Hasil normalisasi data

3.3 Pembagian Data

Dataset yang telah diproses dibagi menjadi dua bagian, yaitu:

1. Data pelatihan (training)

```
# Pembagian data menjadi data train dan data temp
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
print('Data Training: ', X_train.shape, y_train.shape)
print('Data Sementara: ', X_temp.shape, y_temp.shape)
```

```
Data Training:  (1546499, 43) (1546499,)
Data Sementara:  (662786, 43) (662786,)
```

Gambar 13 Pembagian data training dan data sementara

Gambar 13 merupakan gambar untuk membagi *dataset* menjadi *data train* (latih) dan *data temp* (sementara) yang mana 30% data akan digunakan sebagai data sementara sedangkan 70% digunakan untuk melatih data. Dapat dilihat bahwa data train berjumlah 1546499 dan pada data temp berjumlah 662786 data.

2. Data pengujian (testing)

```
# Pembagian data menjadi data validasi dan data test
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
print('Data Validasi: ', X_val.shape, y_val.shape)
print('Data Testing: ', X_test.shape, y_test.shape)
```

```
Data Validasi:  (331393, 43) (331393,)
Data Testing:  (331393, 43) (331393,)
```

Gambar 14 Pembagian data validasi dan testing

Gambar 14 merupakan gambar untuk melakukan pembagian data validasi dan data test. Data sementara yang berjumlah 662.786 data atau 30% dari dataset akan dilakukan pembagian data kembali. Pembagian data ini akan membagi data latih menjadi 50% dan membagi data validasi menjadi 50%. Tujuan melakukan validasi data ini yaitu untuk memastikan model dapat digunakan pada data baru dan dapat meminimalisir ovefitting.

3.4 Pelatihan Model Machine Learning

Algoritma yang digunakan adalah Random Forest, yang merupakan metode ensemble learning berbasis decision tree. Algoritma ini dipilih karena kemampuannya dalam menangani data berdimensi tinggi dan menghasilkan akurasi yang tinggi. Model dilatih menggunakan data pelatihan dan kemudian dievaluasi menggunakan data pengujian untuk menghitung metrik performa seperti akurasi, presisi, recall, dan F1-score.

 ISSN:2302-8149
e-ISSN:2540-9719

```
# Melatih model random forest
param_grid_rf = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2', None],
    'bootstrap': [True, False]
}

rf_model = RandomForestClassifier(random_state=42)
random_search_rf = RandomizedSearchCV(rf_model, param_grid_rf, cv=5, scoring='accuracy', verbose=2, n_
random_search_rf.fit(X_train_under, y_train_under)

best_rf_model = random_search_rf.best_estimator_
print("Best Parameters (Random Forest):", random_search_rf.best_params_)
```

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits
Best Parameters (Random Forest): {'n_estimators': 100, 'min_samples_split': 5, 'min_samples_leaf': 2,
'max_features': 'log2', 'max_depth': 30, 'bootstrap': False}
```

Gambar 15 Melatih model machine learning

Gambar 15 menunjukkan proses pelatihan model *Random Forest* menggunakan *RandomizedSearchCV* untuk menemukan kombinasi parameter terbaik dengan validasi silang sebanyak lima lipatan dan menggunakan metrik akurasi. Model dibangun dengan RandomForestClassifier dari pustaka scikit-learn, dengan parameter random_state=42 untuk menjamin replikasi hasil. Parameter yang dioptimasi meliputi jumlah pohon (n_estimators), kedalaman pohon (max_depth), jumlah minimum sampel untuk pemisahan simpul (min_samples_split), jumlah minimum sampel di daun (min_samples_leaf), jumlah fitur (max_features), serta opsi bootstrap. Model kemudian dilatih menggunakan data pelatihan X_train dan label y_train. Evaluasi model dilakukan menggunakan validasi silang dan klasifikasi report seperti Gambar 16 berikut.

```
# Validasi silang
scores = cross_val_score(best_rf_model, X_train_under, y_train_under, scoring='accuracy', cv=10)
for i, score in enumerate(scores, 1):
    print(f'Fold {i}: {score:.4f}')

# Evaluasi pada data test
y_pred_rf = best_rf_model.predict(X_test)
print("\nRata-rata Akurasi: {score.mean():4f}")
print(f"Evaluasi Model Random Forest:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_rf) * 100:.2f}%")
print(classification_report(y_test, y_pred_rf, target_names=label_encoder.classes_))
```

Gambar 16 Evaluasi model

 ISSN:2302-8149
e-ISSN:2540-9719

```
Fold 1: 0.9989
Fold 2: 1.0000
Fold 3: 0.9989
Fold 4: 0.9978
Fold 5: 0.9989
Fold 6: 1.0000
Fold 7: 1.0000
Fold 8: 0.9989
Fold 9: 0.9989
Fold 10: 1.0000

Evaluasi Model Random Forest:
Accuracy: 99.95%
              precision    recall  f1-score   support

      BENIGN       1.00      1.00      1.00     79585
    PortScan       1.00      1.00      1.00     13885
 SSH-Patator       0.88      1.00      0.93       498
         Syn       1.00      1.00      1.00    237425

    accuracy                           1.00    331393
   macro avg       0.97      1.00      0.98    331393
weighted avg       1.00      1.00      1.00    331393
```

Gambar 17 Hasil evaluasi

```
# Menyimpan Model
joblib.dump(best_rf_model, 'D:\project\model.pkl')

['D:\\project\\model.pkl']
```

Gambar 18 Menyimpan model machine learning

Gambar 17 menampilkan hasil evaluasi model melalui validasi silang 10 lipatan, dengan setiap lipatan (fold) menunjukkan skor akurasi tinggi, yaitu antara 99,89% hingga 100%. Evaluasi dilakukan menggunakan metrik akurasi, presisi, recall, dan F1-score untuk masing-masing kelas: BENIGN, PortScan, SSH-Patator, dan Syn. Hasil menunjukkan bahwa model memiliki presisi dan recall sebesar 1.00 untuk hampir semua kelas, kecuali kelas SSH-Patator yang memiliki presisi 0.88 dan recall 0.93. Akurasi tinggi ini dicapai karena algoritma *Random Forest* mampu menangani banyak fitur dan data kompleks dengan pendekatan *ensemble learning* yang menggabungkan hasil dari banyak *decision tree*, sehingga meningkatkan generalisasi dan mengurangi *overfitting*. Selain itu, proses *preprocessing* yang mencakup penyeimbangan kelas, normalisasi data, dan seleksi fitur berkontribusi terhadap kestabilan *model*. Analisis terhadap hasil akurasi menunjukkan bahwa kelas *SSH-Patator* memiliki karakteristik lalu lintas yang lebih mirip dengan kelas lainnya, sehingga sedikit menurunkan presisi. Namun secara keseluruhan, metrik "macro avg" dan "weighted avg" menunjukkan performa model yang sangat baik dan konsisten pada semua kelas. Gambar 18 memperlihatkan proses penyimpanan model ke dalam file menggunakan pustaka joblib dalam format .pkl (pickle), yang selanjutnya digunakan dalam implementasi sistem IPS. Namun demikian, karena sistem ini digunakan dalam konteks keamanan jaringan, bahkan kesalahan kecil dapat berdampak signifikan. Misalnya, masih adanya ketidaksempurnaan dalam deteksi serangan *SSH-Patator* mengindikasikan bahwa beberapa pola serangan dapat tidak terdeteksi sepenuhnya. Untuk mengurangi risiko ini, sistem dapat dikombinasikan dengan mekanisme pemantauan tambahan atau *model* pembelajaran lanjutan seperti *deep learning* atau *stacking model*. Selain itu, *administrator* tetap disarankan melakukan pemantauan berkala pada jenis serangan yang menunjukkan metrik deteksi yang lebih rendah, guna memastikan perlindungan yang optimal.

3.5 Implementasi Intrusion Prevention System (IPS)

Model Random Forest yang telah dilatih kemudian diintegrasikan ke dalam sistem IPS berbasis Python. Sistem ini berfungsi untuk:

 ISSN:2302-8149
e-ISSN:2540-9719

1. Mendeteksi serangan secara real-time berdasarkan alur lalu lintas jaringan yang dianalisis.
2. Melakukan pemblokiran otomatis terhadap IP sumber serangan menggunakan perintah iptables.
3. Mengirimkan notifikasi otomatis ke admin melalui bot Telegram setiap kali serangan berhasil dideteksi dan diblokir.

Berikut adalah potongan kode program yang dibutuhkan pada IPS:

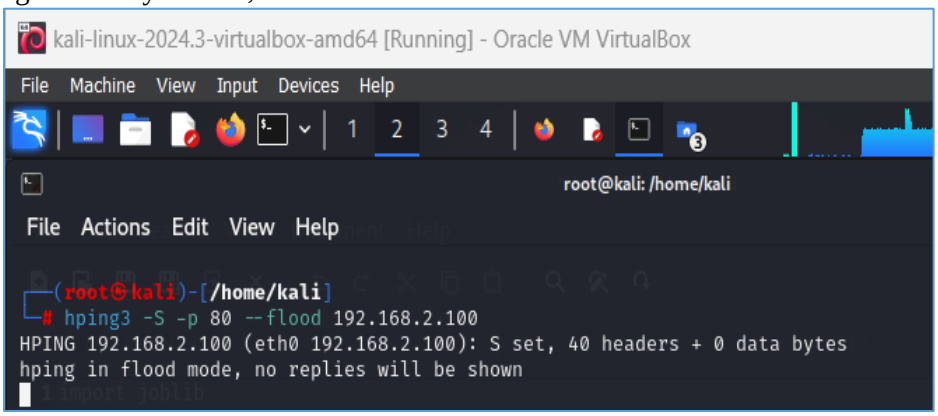
```
import joblib
import pandas as pd
import scapy.all as scapy
import subprocess
import time
import requests
import logging
import statistics
from threading import Thread
from collections import defaultdict
import socket
import warnings
from datetime import datetime
TELEGRAM_TOKEN =
"7444992376:AAFGZS9vqY-zyqTBtFYGuDI8NgUi03AhmnY"
TELEGRAM_CHAT_ID = "5586930689"
model = joblib.load('/home/server/skripsi/model.pkl')
FEATURES = [
    'Source Port', 'Destination Port', 'Protocol', 'Fwd Packet Length Max', 'Fwd
Packet Length Min',
    'Fwd Packet Length Mean', 'Fwd Packet Length Std', 'Bwd Packet Length Max',
'Bwd Packet Length Min',
    'Bwd Packet Length Mean', 'Bwd Packet Length Std', 'Flow Packets/s', 'Bwd
IAT Total', 'Fwd Packets/s',
    'Min Packet Length', 'Max Packet Length', 'Packet Length Mean', 'Packet
Length Std', 'Packet Length Variance',
    'PSH Flag Count', 'ACK Flag Count', 'URG Flag Count', 'Down/Up Ratio',
'Average Packet Size', 'Avg Fwd Segment Size',
    'Avg Bwd Segment Size', 'Init_Win_bytes_backward', 'Inbound', 'RST Flag
Count', 'Unique Destination Ports',
    'SYN/ACK Ratio', 'Attack Duration'
]
warnings.filterwarnings("ignore", category =
UserWarning, module="sklearn")
logging.basicConfig(level=logging.INFO,       format='%(asctime)s       -
%(levelname)s - %(message)s')
IP_ yang_dikecualikan = set()
blocked_ips = set()
ip_flow_stats = defaultdict(lambda: {
    'start_time': time.time(), 'packet_count': 0, 'total_length': 0,
       'bwd_packets':   0,   'unique_ports':   set(),   'syn_count':   0,
'ack_count': 0,
    'failed_logins': 0
})
def get_ips_device():
    hostname = socket.gethostname()
    return socket.gethostbyname(hostname)
IP_ yang_dikecualikan.add(get_ips_device())
```

 ISSN:2302-8149
e-ISSN:2540-9719

3.6 Simulasi Serangan dan Pengujian Sistem

Pengujian dilakukan di lingkungan jaringan lokal dengan melakukan serangan menggunakan tools seperti:

1. hping3 untuk Syn Flood,

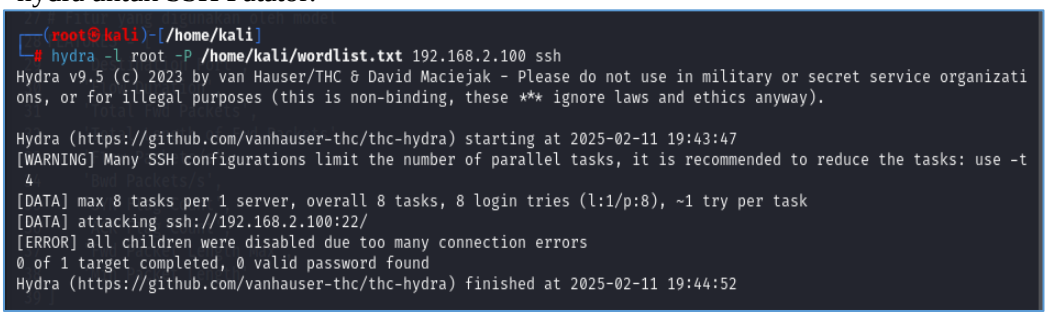


Gambar 19 Komputer penyerang syn flood

2. nmap untuk Port Scanning,

Gambar 20 Komputer penyerang port scanning

3. hydra untuk SSH Patator.



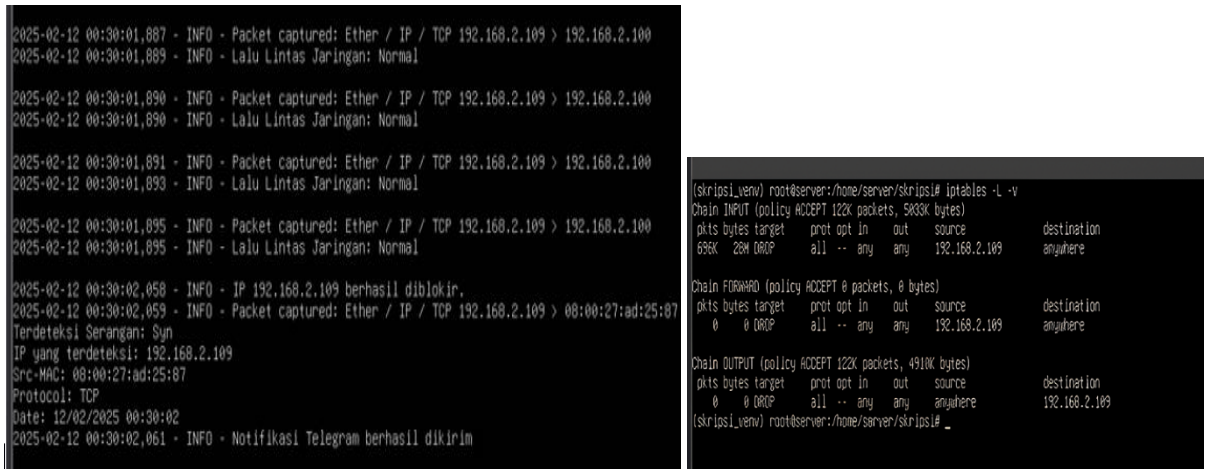
Gambar 21 Komputer Penyerang SSH Patator

Selama simulasi berlangsung, sistem IPS aktif melakukan monitoring, deteksi, dan penindakan otomatis. Pengujian dilakukan untuk mengukur ketepatan deteksi, waktu respon, serta validitas notifikasi.

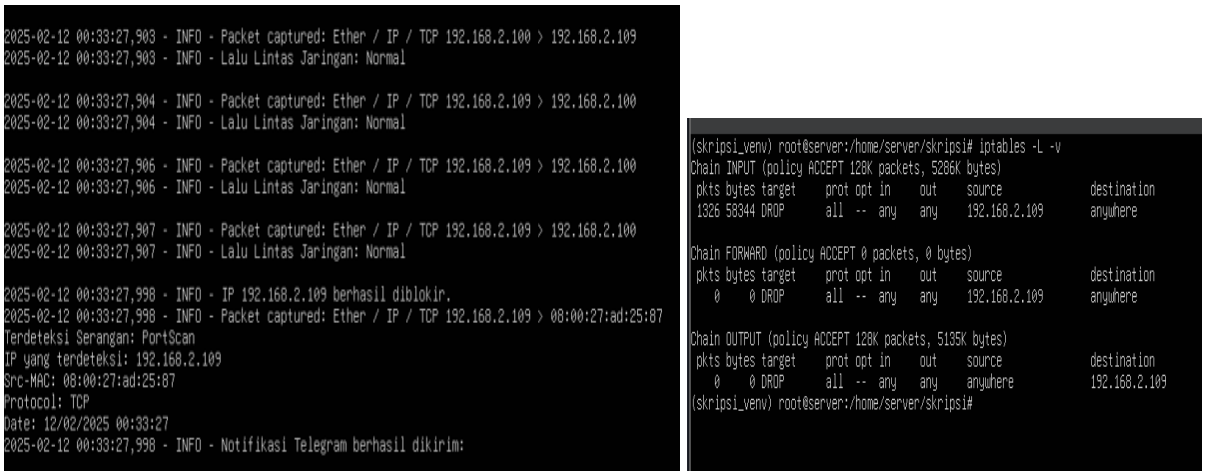
4 Hasil dan Pembahasan

Hasil dari implementasi serta pengujian *Intrusion Prevention System* (IPS) yang telah dilakukan dan dikembangkan menggunakan model *machine learning* berbasis *Random Forest*. Fokus pengujian berada pada kemampuan sistem dalam mendeteksi serangan jaringan secara *real-time*, memblokir alamat IP penyerang, serta mengirimkan notifikasi otomatis ke Telegram. Hasil ditampilkan dalam bentuk dokumentasi proses deteksi dan bukti pengiriman notifikasi, yang menggambarkan respons sistem terhadap jenis serangan yang diuji. Pembahasan difokuskan pada efektivitas sistem dalam merespons ancaman secara otomatis dan akurat.

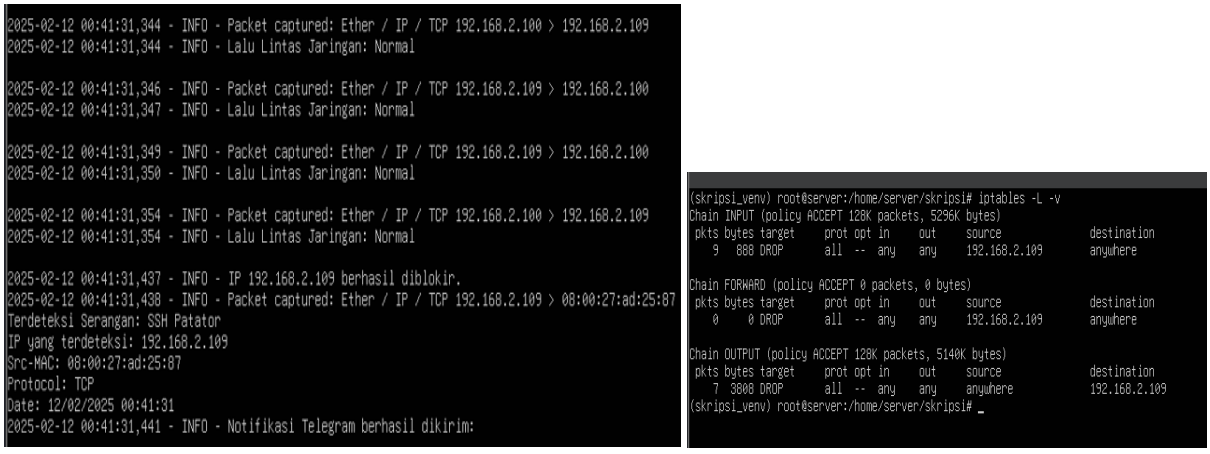
 ISSN:2302-8149
e-ISSN:2540-9719



Gambar 22 Sistem IPS mendeteksi serangan syn flood



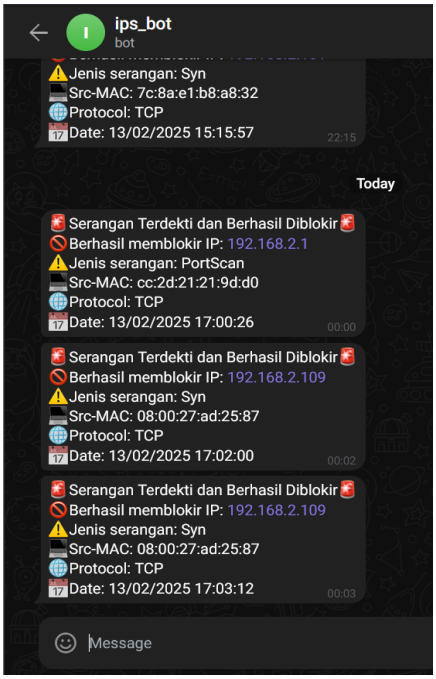
Gambar 23 Sistem IPS mendeteksi serangan port scanning



Gambar 24 Sistem IPS mendeteksi serangan SSH patator

Pada Gambar 22, Gambar 23 dan Gambar 24 dapat dilihat bahwa hasil pengujian sistem IPS berdasarkan serangan *syn flood, port scanning dan ssh patator* berhasil terdeteksi. Pada saat serangan terdeteksi, sistem ips secara otomatis menampilkan ip penyerang, mac-address, protocol, waktu serta mengirimkan notifikasi ke telegram bahwa ip penyerang sudah berhasil diblokir.

 ISSN:2302-8149
e-ISSN:2540-9719



Gambar 25 Hasil blokir real-time

Gambar 25 menunjukkan keberhasilan implementasi sistem IPS yang dilengkapi dengan notifikasi otomatis melalui Telegram setelah berhasil memblokir alamat IP penyerang. Notifikasi yang dikirimkan berisi informasi penting seperti alamat IP penyerang, jenis serangan, alamat MAC, protokol, serta waktu kejadian, yang sangat membantu dalam pengawasan insiden keamanan secara real-time. Sistem ini menggunakan metode manual polling untuk berinteraksi langsung dengan API Telegram, memungkinkan pengiriman notifikasi yang cepat dan responsif. Meskipun metode ini membutuhkan koneksi jaringan yang stabil, keunggulan dari pendekatan ini terletak pada kemampuannya memberikan pemberitahuan instan tanpa keterlambatan. Dengan fitur ini, sistem IPS yang dikembangkan dalam penelitian ini menawarkan nilai tambah signifikan dibandingkan pendekatan konvensional, karena memungkinkan administrator mengetahui ancaman secara langsung tanpa perlu memeriksa log server secara manual, sehingga meningkatkan efisiensi dan kecepatan respons terhadap insiden keamanan jaringan.

5 Kesimpulan

Penelitian ini berhasil mengembangkan sistem Intrusion Prevention System (IPS) berbasis machine learning dengan algoritma Random Forest yang mampu mendeteksi dan mencegah tiga jenis serangan jaringan secara otomatis, yaitu Syn Flood, Port Scanning, dan SSH Patator. Model yang dilatih menggunakan dataset gabungan dari CICIDS2017 dan CICIDS2019 menunjukkan performa evaluasi yang tinggi, dengan akurasi mencapai hampir 100% dalam validasi silang. Sistem tidak hanya mampu mendeteksi serangan secara real-time, tetapi juga melakukan pemblokiran otomatis terhadap IP penyerang dan mengirimkan notifikasi langsung ke Telegram, yang sangat membantu administrator jaringan dalam merespons ancaman dengan cepat tanpa harus melakukan pemantauan manual. Dengan pendekatan ini, sistem IPS yang dikembangkan dalam penelitian ini menunjukkan keunggulan dalam hal otomatisasi, efisiensi operasional, dan akurasi deteksi, serta memberikan kontribusi nyata dalam peningkatan keamanan jaringan yang bersifat adaptif dan responsif. Sebagai arah pengembangan ke depan, sistem ini masih memiliki potensi untuk ditingkatkan, khususnya dalam mendeteksi jenis serangan lainnya seperti SQL Injection, ICMP Flood, atau serangan berbasis aplikasi yang kompleks. Penelitian lanjutan juga dapat mempertimbangkan penggunaan algoritma lain seperti deep learning atau hybrid model untuk meningkatkan akurasi dan cakupan deteksi dalam menghadapi skenario serangan nyata yang lebih beragam.

 ISSN:2302-8149
e-ISSN:2540-9719

Referensi

[1] R. A. Azmi, K. Rukun, and H. Maksum, "Analisis Kebutuhan Pengembangan Media Pembelajaran berbasis *Web* Mata Pelajaran Administrasi Infrastruktur Jaringan," *JIPP*, Vol. 4, Jul. 2020.

[2] R. E. Susanti, A. W. Muhammad, and W. A. Prabowo, "Implementasi *Intrusion Prevention System (IPS) OSSEC* dan *Honeypot Cowrie*," *Jurnal Sisfokom (Sistem Informasi dan Komputer)*, Vol. 11, No. 1, pp. 73–78, Apr. 2022, DOI: 10.32736/sisfokom.v11i1.1246.

[3] Nuroji, "Penerapan *Intrusion Detection and Prevention System* (IDPS) pada Jaringan Komputer sebagai Pencegahan Serangan *Port-Scanning*," *Journal of Data Science and Information System (DIMIS)*, Vol. 1, pp. 41–49, May 2023, DOI: 10.58602/dimis.v1i2.35.

[4] H. Awal and A. P. Gusman, "Implementasi *Intrusion Detection Prevention System* sebagai Sistem Keamanan Jaringan Komputer Kejaksaan Negeri Pariaman menggunkan *SNORT* dan *Iptable*s berbasis *Linux*," *Jurnal Sains Informatika Terapan (JSIT) E-ISSN*, Vol. 2, No. 2, pp. 74–80, Jun. 2023.

[5] R. Kurniawan and F. Prakoso, "Implementasi Metode IPS (*Intrusion Prevention System*) dan IDS (*Intrusion Detection System*) untuk meningkatkan Keamanan Jaringan," *Jurnal SENTINEL*, Vol. 2, No. 02, pp. 231–242, Jan. 2020.

[6] T. Prasetyo, "Pengamanan Jaringan Komputer dengan *Intrusion Prevention System* (IPS) berbasis SMS *Gateway*," Vol. 2, pp. 1–13, Jun. 2022.

[7] B. Kriswantara and R. Sadikin, "*Used Car Price Prediction with Random Forest Regressor Model*," *Journal of Information Systems, Informatics and Computing Issue Period*, Vol. 6, No. 1, pp. 40–49, Jun. 2022, DOI: 10.52362/jisicom.v6i1.752.

[8] L. I. Uzlah, R. A. Saputra, and Isnawaty, "Deteksi Serangan Siber pada Jaringan Komputer menggunakan Metode *Random Forest*," *Jurnal Mahasiswa Teknik Informatika*, Vol. 8, No. 3, Jun. 2024, [Online]. Available: https://bit.ly/CyberSecurityAttacks.

[9] A. Anggraeni, J. G. A. Ginting, and S. Ikhwan, "*Implementation of Intrusion Prevention System* (IPS) *to Analysis Triad Cia on Network Security Attacks on Web Server*," *Jurnal Infotel*, Vol. 14, No. 4, pp. 277–286, Nov. 2022, DOI: 10.20895/infotel.v14i4.813.

[10] J. K. Barends, F. Dewanta, and N. B. A. Karna, "Perancangan dan Analisis *Intrusion Prevention* Sistem berbasis *SNORT* dan *IPTABLES* dengan *Integrasi Honeypot* pada Arsitektur *Software Defined Network*," *Jurnal Multinetics*, Vol. 7, No. 2, pp. 163–176, Nov. 2021.

[11] Mr. S. Waskle, Mr. L. Parashar, and Mr. U. Singh, "*Intrusion Detection System using PCA with Random Forest Approach*," in *Proceedings of the International Conference on Electronics and Sustainable Communication Systems (ICESC 2020)*, 2020.

[12] M. Thariq and S. Rendratama, "Perancangan dan Analisis *Crowdsec* sebagai *Intrusion Prevention System* pada Infrastruktur *Server*," Vol. 10, No. 2, pp. 1887–1894, Apr. 2023.

[13] Farhannullah and M. Hardjianto, "Sistem Monitoring Serangan Ssh dengan Metode *Intrusion Prevention System* (IPS) Fail2ban menggunakan *Python* pada Sistem Operasi Linux," *Jurnal TICOM: Technology of Information and Communication*, Vol. 11, No. 1, pp. 33–38, Sep. 2022.

[14] A. Kurniawan and L. M. Silalahi, "Analisis Keamanan Jaringan menggunakan *Intrusion Prevention System* (IPS) dengan Metode *Traffic Behavior*," *ELECTRICIAN – Jurnal Rekayasa dan Teknologi Elektro*, Vol. 17, No. 1, pp. 71–76, Jan. 2023.

[15] S. Goutama, A. Noertjahyana, and H. N. Palit, "Simulasi Aplikasi untuk mendeteksi dan mencegah Serangan DDoS pada Jaringan berbasis *Software Defined Network*," *Jurnal Infra*, Vol. 10, No. 1 (2022), Jan. 2022.