Penggunaan Data Kotor Untuk Melatih Sistem Deteksi Serangan Berbasis Jaringan

Krisna Badru Wijaya
Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember (ITS)
e-mail: krisnawijaya.17051@mhs.its.ac.id

***Abstrak*— Keamanan sistem dan jaringan merupakan hal yang sangat penting untuk dijaga oleh setiap perangkat yang terhubung ke jaringan. Kemudahan akses pada jaringan menjadi pintu terbuka untuk pelaku-pelaku kejahatan siber melakukan serangan terhadap perangkat targetnya. Berbagai jenis serangan dapat kita temui secara sadar atau tidak sadar saat mengakses data pada jaringan. Dikarenakan banyaknya jenis data dan koneksi yang terjadi di jaringan, sangat sulit untuk membedakan antara koneksi yang normal dan koneksi yang tidak normal pada jaringan. Untuk mencegah terjadinya kejahatan siber yang dapat membahayakan perangkat pengguna, setiap perangkat yang terhubung pada jaringan harus bisa mengidentifikasi keamanan koneksi yang masuk ke dalam perangkat.**

Sistem deteksi serangan berbasis jaringan atau Network Intrusion Detection System (NIDS) adalah salah satu jenis sistem pendeteksi serangan yang bekerja pada jaringan komputer. NIDS berfungsi untuk memonitor, mengidentifikasi, dan menganalisa lalu lintas jaringan komputer dari potensi serangan pada keamanan jaringan. Data-data koneksi yang masuk pada jaringan akan diperiksa oleh NIDS. Apabila ditemukan keanehan-keanehan saat dilakukannya analisa, NIDS akan langsung menotifikasi host.

Salah satu metode yang digunakan pada NIDS untuk melatih sistem dalam mendeteksi serangan pada jaringan adalah dengan menggunakan algoritma Machine Learning. NIDS dilatih untuk mengenali koneksi-koneksi normal dan koneksi-koneksi anomali yang ada pada jaringan. Normalnya, data latih yang digunakan untuk melatih NIDS adalah data yang sama sekali tidak mengandung unsur serangan. Akan tetapi dalam prakteknya di dunia nyata, data seperti itu sangat sulit untuk didapatkan.

Dalam jurnal kali akan akan dibahas tentang perancangan sistem deteksi serangan cerdas yang bisa mendeteksi serangan dengan menggunakan data kotor atau data campuran yang di dalamnya masih berpotensi mengandung unsur serangan.

Kata Kunci—Machine learning, NIDS, Data Kotor.

I. PENDAHULUAN

Salah satu cara untuk mendeteksi serangan siber pada perangkat yang dilindungi adalah dengan menggunakan sistem deteksi intrusi berbasis jaringan atau *Network Intrusion Detection System* (NIDS). NIDS berfungsi untuk memonitor, mengidentifikasi, dan menganalisa lalu lintas jaringan komputer dari potensi serangan pada keamanan jaringan. Agar NIDS dapat mendeteksi serangan dengan baik, NIDS harus dilatih terlebih dahulu mengenali koneksi-koneksi yang normal dan koneksi-koneksi yang bersifat anomali.

Salah satu cara untuk melatih NIDS adalah dengan menggunakan algoritma *machine learning.* Algoritma *machine learning* dapat melakukan prediksi terhadap koneksi-koneksi yang masuk terhadap perangkat. Agar *machine learning* dapat mengidentifikasi data dengan baik, *machine learning* dilatih menggunakan data latih yang disiapkan terlebih dahulu. Dari data latih tersebut, *machine learning* dapat menentukan pola data yang selanjutnya menjadi model dalam melakukan prediksi data. Model yang terbentuk menjadi patokan saat melakukan prediksi data test. Kinerja algoritma *machine learning* diukur dengan melihat patokan nilai-nilai dari kesuksesan algoritma dalam memprediksi data dengan benar.

Salah satu jenis NIDS adalah *Anomaly-based Intrusion Detection System* (AIDS). AIDS mendeteksi data dengan memperhitungkan nilai-nilai pada data yang memiliki pola tidak normal. Beberapa algoritma yang dapat digunakan untuk melatih AIDS adalah *One-Class SVM, Local Outlier Factor*, dan *Isolation Forest*. Algoritma-algoritma tersebut tidak memerlukan data yang berlabel dalam melakukan prediksi, sehingga memungkinkan untuk menggunakan data kotor untuk melatih algortima tersebut.

Data yang digunakan adalah data yang berformat PCAP. PCAP berisi data paket-paket yang masuk pada protokol TCP/IP dan UDP. Data dapat berupa IP address, port, payload, sumber dan destinasi dari paket yang terdeteksi. Payload adalah konten dari paket yang masuk. Seluruh data payload pada file PCAP akan diproses lagi menjadi data *byte frequency*. *Byte frequency* merupakan data frekuensi *byte* pada masing-masing payload pada setiap paket. Dari analisa *byte frequency* tersebut, kita dapat membuat sebuah model *machine learning* untuk mencari anomali yang terjadi pada data test yang digunakan.

II. TINJAUAN PUSTAKA

A. NIDS

Network Intrusion Detection System (NIDS) atau sistem deteksi serangan berdasarkan jaringan adalah salah satu jenis *Intrusion Detection System (IDS)* yang bekerja untuk memonitor lalu lintas jaringan. NIDS ditempatkan pada titik strategis yang ada pada jaringan. Selanjutnya, NIDS akan melakukan analisis seluruh paket data yang melintas pada subnet NIDS ditempatkan. Paket data yang dianalisa NIDS akan dicocokkan dengan daftar pustaka serangan yang dimiliki NIDS. Bila paket teridentifikasi sebagai serangan, NIDS akan memberikan peringatan ke administrator jaringan [1].

B. Python

Python adalah bahasa pemrograman tingkat tinggi yang dirancang agar mudah dibaca, dan memiliki beberapa kesamaan dengan bahasa inggris. Salah satu keunggulan

python adalah luasnya daftar pustaka yang dimiliki oleh python. Hal ini memudahkan programmer dalam menjalankan fungsi-fungsi kompleks pada program yang dibuat. Selain itu, fitur-fitur dan modul yang ada pada python juga tidak dikenakan biaya dalam penggunaanya. Python dipilih sebagai basis bahasa pemrograman dalam tugas akhir ini karena python memiliki fungsi-fungsi dan daftar pustaka yang mendukung dalam kegiatan analisis data jaringan. Simulasi sniffing dan capture paket menjadi mudah dilakukan dengan menggunakan fungsi-fungsi yang ada [2].

C. *Machine Learning*

Machine learning adalah algoritma komputer yang secara otomatis dapat mengembangkan dirinya berdasar data latih yang diberikan [3]. Untuk menyelesaikan tugas yang diberikan, algoritma *machine learning* membangun model berdasarkan data sampel, yang dikenal sebagai "data latih", untuk membuat prediksi atau keputusan tanpa diprogram secara eksplisit untuk melakukannya. Beberapa jenis *machine learning* berhubungan erat dengan komputer statistik. Dari data hasil statistik, *machine learning* dapat memprediksi output yang sesuai dengan tugas yang diberikan.

D. *Scikit-learn*

Scikit-learn adalah pustaka *machine learning* yang bersifat *open-source*. Scikit-learn membantu pengguna dalam pembuatan algoritma *supervised learning* dan *unsupervised learning*. Scikit-learn juga menyediakan berbagai alat untuk pencocokan model, pra-pemrosesan data, pemilihan dan evaluasi model, dan banyak utilitas lainnya. Didalam pustaka scikit-learn sudah ada *built-in* algoritma *machine learning* yang membantu dalam melakukan pemrosesan data [4].

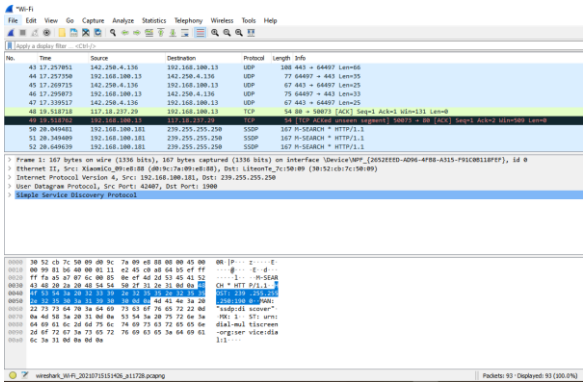
E. *Payload*

Tabel 1. Format Datagram IP

Header Protokol Antarmuka Jaringan	Header Protokol IP	Payload IP	Trailer Protokol Antarmuka jaringan

Paket-paket data dalam protokol IP dikirimkan dalam bentuk datagram. *Network intrusion detection system* dibagi menjadi 2 tipe berdasarkan bagian datagram yang dianalisis yaitu NIDS berdasarkan *header* dan NIDS berdasarkan *payload*. *Payload* merupakan isi atau muatan dari sebuah paket jaringan yang akan ditransmisikan, dimana *payload length* menentukan panjang dari isi data yang dienkapsulasi di dalam paket dalam sebuah *byte*. *[5]*

F. *PCAP*



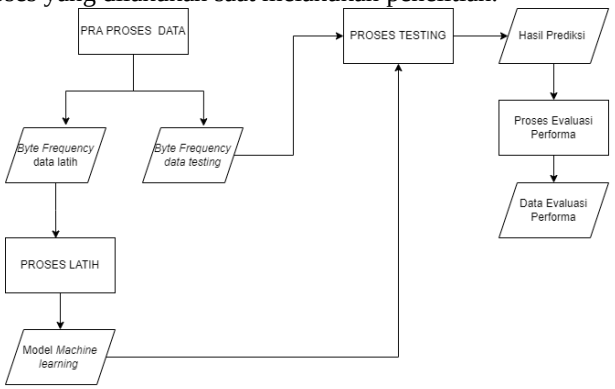
Gambar 1 *Packet capture* dengan aplikasi Wireshark

PCAP (singkatan dari Packet Capture) adalah nama API yang biasa digunakan untuk merekam metrik paket. File PCAP sangat membantu karena dapat digunakan untuk merekam data lalu lintas multilayer, menangkap paket yang berasal dari lapisan data link sampai ke lapisan aplikasi. Data rekaman dari PCAP selanjutnya bisa digunakan untuk kegiatan analisa paket jaringan. Data latih dan data testing NIDS biasanya menggunakan file PCAP.

III. METODOLOGI

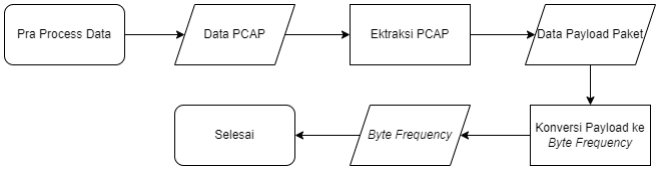
A. *Diagram Sistem*

Pada jurnal ini akan dibuat sebuah sistem deteksi intrusi jaringan yang dilatih menggunakan data yang mengandung data kotor pada algoritma *machine learning*. Sistem menggunakan algoritma *unsupervised learning* dengan membandingkan nilai-nilai anomali pada setiap data latih yang diberikan. Sistem diharapkan mampu memprediksi dengan baik apakah data koneksi termasuk data serangan atau data normal. Berikut merupakan alur diagram penelitian yang menggambarkan sebagian besar proses-proses yang dilakukan saat melakukan penelitian.



Gambar 2 Diagram Alur Penelitian

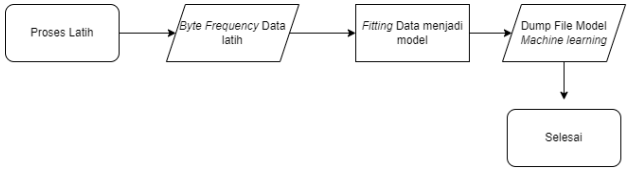
B. *Pra Proses Data*



Gambar 3 Urutan Pelaksanaan Penelitian

Pada pra proses data, terdapat 2 kegiatan yang akan dilakukan untuk memproses file PCAP hingga menjadi *byte frequency*. Proses pertama yang dilakukan adalah ektraksi PCAP. Pada proses ini, sistem akan membaca file PCAP untuk mendapatkan header dan payload dari data PCAP tersebut. Setelah didapatkan data payload, payload akan dikonversikan menjadi *byte frequency*. Masing-masing data latih dan data testing akan dikonversikan menjadi *byte frequency* sebelum digunakan pada proses latih dan proses testing.

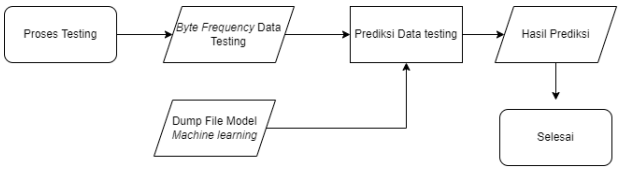
C. *Proses Latih*



Gambar 3 Diagram Alur Proses Latih

Byte frequency data latih yang didapatkan dari hasil pra proses akan digunakan untuk pembuatan model *machine learning* pada proses latih. Data *byte frequency* dipakai dalam proses *fitting*. Dalam proses *fitting*, data *byte frequency* dianalisa hingga menghasilkan sebuah nilai parameter yang digunakan dalam pembuatan model. Nilai parameter ini yang selanjutnya menjadi patokan saat melakukan prediksi data menggunakan model yang telah terbentuk.

D. *Proses Testing*



Gambar 3 Diagram Alur Proses Testing

Ada 2 komponen utama yang diperlukan untuk menjalankan proses testing, yaitu data testing yang sudah diproses menjadi data *byte frequency* dan model yang didapatkan dari proses latih. Data testing yang telah diproses menjadi data *byte frequency* merupakan data mentah yang belum memiliki label untuk membedakan apakah data tersebut merupakan data serangan atau data normal. Sistem akan melakukan prediksi untuk memberikan pelabelan pada masing-masing data latih. Apabila sistem menganggap data tersebut adalah data yang terindentifikasi serangan, sistem akan memberikan label -1 (*outlier)*. Data yang terindentifikasi sebagai data normal akan diberikan label 1 (*inlier).*

E. *Evaluasi Performa Algoritma*

Evaluasi uji coba dilakukan dengan melihat hasil dari prediksi data testing. Hasil prediksi data testing disimpulkan menjadi *confusion matrix. Confusion matrix* digunakan untuk membandingkan hasil prediksi yang dilakukan oleh program dengan hasil prediksi yang benar. Dari *confusion matrix* kita dapat mendapatkan nilai *true positive, true negative, false positive*, dan *false negative* dari hasil prediksi. Berikut merupakan tabel penjelasan dari confusion matrix:

Tabel 2. Penjelasan *Confusion Matrix*

No	Nama	Penjelasan
1	*True Positive (TP)*	Hasil prediksi mendeteksi sebagai *outlier*, dan hasil aktual menandakan data *malicious.*
2	*True Negative* (TN)	Hasil prediksi mendeteksi sebagai *inlier*, dan hasil aktual menandakan data tidak *malicious*
3	*False Positive* (FP)	Hasil prediksi mendeteksi sebagai *outlier*, sedangkan hasil aktual menandakan data tidak *malicious*
4	*False Negative* (FN)	Hasil prediksi mendeteksi sebagai *inlier*, sedangkan hasil aktual menandakan data *malicious*

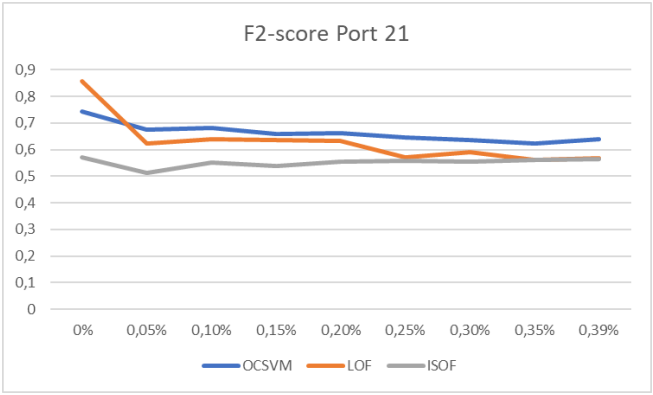
IV. HASIL DAN PEMBAHASAN

A. *Skenario Evaluasi Percobaan*

Percobaan dibedakan menjadi 3 jenis berdasarkan protokol masing-masing dataset. Setiap percobaan dilakukan sebanyak 3 kali perulangan dengan 3 algoritma *machine learning* yang berbeda. Pada subbab ini akan dipaparkan data hasil percobaan dan analisa berdasarkan nilai F_2*-score* masing-masing algoritma

B. *Hasil Pengujian Protokol FTP*

Pada protokol FTP terdapat total 9 data latih PCAP dengan kadar yang berbeda-beda. Masing-masing data PCAP memiliki ukuran dengan rata-rata 228 MB. Pengujian dilakukan sebanyak 3 kali percobaan pada setiap 3 algoritma yang berbeda. Tiap algoritma menggunakan parameter *default* dari pustaka sckit-learn. Hasil total terdapat 81 data *confusion matrixes.* Proses eksekusi program membutuhkan waktu rata-rata 10 menit dari proses latih hingga proses testing. Berikut merupakan visualisasi nilai F_2*-score* dari data hasil pengujian:



Gambar 4 Grafik F_2*-score* Hasil Pengujian FTP

Dari data diatas dapat kita lihat nilai puncak F_2*-score* pada masing-masing data terdapat pada saat dilakukan percobaan pada dataset dengan kadar 0%. Penurunan F_2*-score* tampak jelas terlihat pada grafik seiring dengan kenaikan kadar data kotor pada data yang digunakan. Pada algoritma *One-Class SVM* puncak tertinggi terjadi pada kadar 0% dengan nilai F_2*-score* sebesar 0,74. Lalu terjadi penurunan nilai F_2*-score* seiring dengan menaiknya kadar data kotor dengan rentang nilai 0,68-0,62, puncak terendah terjadi pada data yang memiliki kadar 0.35% dengan nilai F_2*-score* sebesar 0.62.

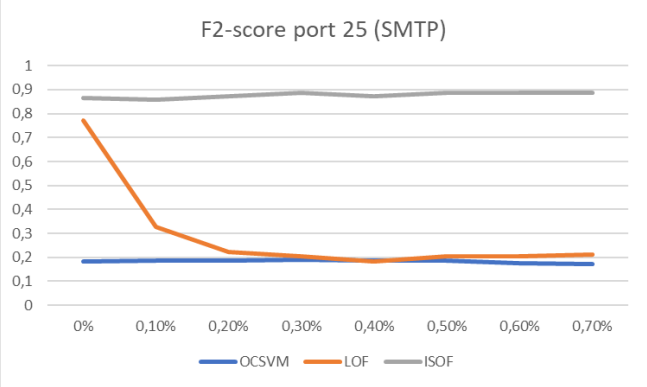
Pada algoritma *Local Outlier Factor*, puncak tertinggi nilai F_2*-score* juga terjadi pada data yang memiliki kadar

0% dengan nilai F_2-*score* sebesar 0,85. Dari nilai F_2-*score* sebesar 0,85, nilai F_2-*score* data pada algoritma LOF mengalami penurunan dengan titik terendah pada nilai 0,5602 pada data dengan kadar data kotor sebesar 0.35%.

Sedangkan pada algoritma *Isolation Forest*, puncak tertinggi memiliki nilai F_2-*score* sebesar 0,57 yang terjadi pada data dengan kadar data kotor sebesar 0%. Puncak terendah terjadi pada data dengan kadar data kotor sebesar 0,05% dengan nilai F_2-*score* sebesar 0,51.

C. *Hasil Pengujian Protokol SMTP*

Pada protokol SMTP terdapat total 8 data latih PCAP dengan kadar yang berbeda-beda. Masing-masing data PCAP memiliki ukuran dengan rata-rata 1,9 GB. Pengujian dilakukan sebanyak 3 kali percobaan pada setiap 3 algoritma yang berbeda. Tiap algoritma menggunakan parameter *default* dari pustaka sckit-learn. Hasil total terdapat 72 data *confusion matrixes*. Proses eksekusi program membutuhkan waktu rata-rata 1 jam 30 menit dari proses latih hingga proses testing. Berikut merupakan visualisasi nilai F_2-*score* dari data hasil pengujian:



Gambar 4 Grafik F_2-*score* Hasil Pengujian SMTP

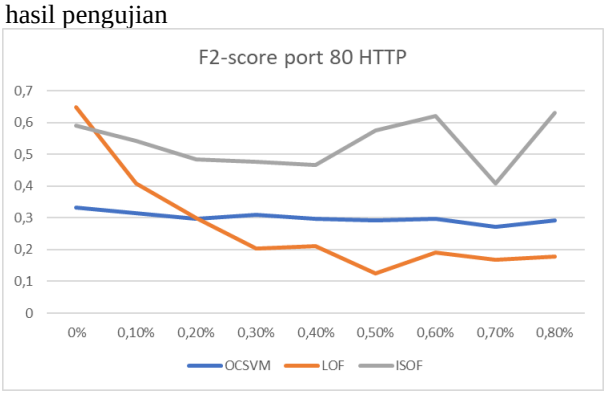
Pada algoritma *Isolation Forest* tampak nilai F_2-*score* relatif konstan berkisar pada rentang 0,85 – 0,88. Perbedaan F_2-*score* tidak terlalu signifikan dengan selisih batas atas dan batas bawah hanya 0,03.

Pada algoritma *Local Outlier Factor*, titik puncak terjadi pada data dengan kadar data kotor sebesar 0% dengan nilai F_2-*score* sebesar 0,77. Namun, penurunan nilai F_2-*score* tampak sangat jelas dibandingkan algoritma *One-Class SVM*. Dari angka 0,77, nilai F_2-*score* turun drastis menjadi 0,32. Titik terendah terjadi pada data testing dengan kadar data sebesar 0,4% dengan nilai F_2-*score* sebesar 0,18.

Sedangkan pada algoritma *One-Class SVM*, tampak nilai F_2-*score* relatif stabil tetapi rata-rata nilainya terlalu rendah dibanding dengan algoritma lain. Nilai F_2-*score* hanya berkisar pada rentang 0,17-0,18. Sangat rendah dibanding dengan algoritma yang lain. Titik tertinggi terjadi pada data testing dengan kadar 0,6%. Dan titik terendah terjadi pada data testing dengan kadar 0,7%.

D. *Hasil Pengujian Protokol HTTP*

Pada protokol SMTP terdapat total 9 data latih PCAP dengan kadar yang berbeda-beda. Masing-masing data PCAP memiliki ukuran dengan rata-rata 11,4 GB. Pengujian dilakukan sebanyak 3 kali percobaan pada setiap 3 algoritma yang berbeda. Tiap algoritma menggunakan parameter *default* dari pustaka sckit-learn. Hasil total terdapat 81 data *confusion matrix*. Proses eksekusi program membutuhkan waktu rata-rata 10 jam dari proses latih hingga proses testing. Berikut merupakan visualisasi nilai F_2-*score* dari data hasil pengujian



Gambar 5 Grafik F_2-*score* Hasil Pengujian HTTP

Pada algoritma *One-Class SVM*, tampak nilai F_2-*score* relatif stabil pada kisaran 0,33 – 0,27. Puncak tertinggi terjadi pada data dengan kadar data kotor sebesar 0% dengan nilai F_2-*score* sebesar 0,33. Sedangkan puncak terendah terjadi pada data dengan kadar data kotor sebesar 0,7% dengan nilai F_2-*score* sebesar 0,27.

Pada algoritma *Local Outlier Factor*, titik puncak terjadi pada data dengan kadar data kotor sebesar 0% dengan nilai F_2-*score* sebesar 0,64. Tampak nilai F_2-*score* mengalami penuruan drastis seiring bertambahnya kadar data kotor dengan penuruan dari 0,64 sampai dengan nilai puncak terendah sebesar 0,12 yang terjadi pada data dengan kadar data kotor sebesar 0,05%.

Sedangkan pada algoritma *Isolation Forest*, tampak nilai bergelombang dengan rentan nilai F_2-*score* antara 0,63-0,40. Nilai puncak terjadi pada data testing dengan kadar data kotor sebesar 0,8%. Sedangkan puncak terendah terjadi pada data testing dengan kadar data kotor sebesar 0,7%.

V. KESIMPULAN/RINGKASAN

Setelah dilakukan percobaan terdapat beberapa kesimpulan yang dapat dipetik. Peningkatan kadar data kotor pada data testing tampak mempengaruhi performa dari beberapa algoritma *machine learning*. Seperti saat dilakukan percobaan pada port 21, tampak ada penurunan nilai F_2-*score* dari data testing dengan kadar 0% terhadap data testing yang lain. Nilai F_2-*score* cenderung turun seiring naiknya tingkat kadar data kotor.

Namun, tidak semua percobaan tampak demikian. Ada beberapa algoritma *machine learning yang relatif stabil*, meski kadar data kotor yang ada meningkat. Ada pula yang tidak stabil dan bersifat bergelombang sehingga tidak dapat diprediksi dengan pasti.

DAFTAR PUSTAKA

[1] Leonid Portnoy, “Intrusion detection with unlabeled data using clustering” Departemen of Computer Science Columbia University, 2001.

[2] “Python Language advantages and applications” GeeksforGeeks, 30 Juni 2021. [Online] Available: https://www.geeksforgeeks.org/python-language-advantages-applications/ [Accessed 22 Juni 2021].

[3] Mitchell, Tom (1997). “Machine Learning”. New York: McGraw Hill. ISBN 0-07-042807-7. OCLC 36417892..

[4] “Getting started scikit-learn” Scikit-learn. [Online] Available: https://scikit-learn.org/stable/getting_started.html [Accessed 25 Juni 2021].
[5] Bangun, Chandra Sampe, “Uji Perbandingan Sistem Deteksi Intrusi Berdasarkan Sumber Data Header dan Payload” Program Studi Teknik Informatika FTI-UKSW, 2013.