IDENTIFIKASI ANOMALI LALU LINTAS JARINGAN MELALUI KOMBINASI *LOCAL OUTLIER FACTOR* (LOF) DAN *RULE-BASED SYSTEM*

SKRIPSI

RISKI HARTANTO SARAGIH
201402112

PROGRAM STUDI S1 TEKNOLOGI INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2025

IDENTIFIKASI ANOMALI LALU LINTAS JARINGAN MELALUI KOMBINASI *LOCAL OUTLIER FACTOR* (LOF) DAN *RULE-BASED SYSTEM*

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah Sarjana Teknologi Informasi

RISKI HARTANTO SARAGIH

201402112

PROGRAM STUDI S1 TEKNOLOGI INFORMASI

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

UNIVERSITAS SUMATERA UTARA

MEDAN

2025

PERSETUJUAN

Judul	: IDENTIFIKASI ANOMALI LALU LINTAS JARINGAN MELALUI KOMBINASI *LOCAL OUTLIER FACTOR* (LOF) DAN *RULE-BASED SYSTEM*
Kategori	: SKRIPSI
Nama	: RISKI HARTANTO SARAGIH
Nomor Induk Mahasiswa	: 201402112
Program Studi	: SARJANA (S1) TEKNOLOGI INFORMASI
Fakultas	: ILMU KOMPUTER DAN TEKNOLOGI INFORMASI UNIVERSITAS SUMATERA UTARA

Medan, 6 Maret 2025

Komisi Pembimbing

Pembimbing 2	Pembimbing 1
Dr. Romi Fadillah Rahmat, B.Comp.Sc., M.Sc.	Ainul Hizriadi, S.Kom., M.Sc
NIP : 198603032010121004	NIP : 198510272017061001

Diketahui/disetujui oleh

Program Studi Teknologi Informasi

Ketua,

Dedy Arisandi, S.T., M.Kom.

NIP : 197908312009121002

PERNYATAAN

IDENTIFIKASI ANOMALI LALU LINTAS JARINGAN MELALUI KOMBINASI *LOCAL OUTLIER FACTOR* (LOF) DAN *RULE-BASED SYSTEM*

SKRIPSI

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, 6 Maret 2025

Riski Hartanto Saragih
201402112

UCAPAN TERIMA KASIH

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa dengan limpah karunia-Nya penulis dapat menyelesaikan penyusunan tugas akhir ini dengan judul "Identifikasi Anomali Lalu Lintas Jaringan melalui Kombinasi *Local Outlier Factor* (LOF) dan *Rule-Based System*". Penyusunan tugas akhir ini dimaksudkan untuk memenuhi persyaratan mencapai gelar akademik Sarjana Komputer pada Program Studi S1 Teknologi Informasi, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara.

Penulis menyadari bahwa tugas akhir ini tidak dapat terselesaikan tanpa bantuan, dukungan, bimbingan dan masukan berupa kritik serta saran dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis mengucapkan terima kasih kepada semua pihak yang telah membantu dalam penyusunan tugas akhir ini terutama kepada:

1. Kedua orang tua penulis, alm. Ayah J. Saragih dan Ibu L. R. Purba tercinta yang senantiasa mendoakan, mendukung, serta menasehati penulis.
2. Bapak Ainul Hizriadi, S.Kom., M.Sc selaku Dosen Pembimbing 1 dan Bapak Dr. Romi Fadillah Rahmat, B.Comp.Sc., M.Sc. selaku Dosen Pembimbing 2 yang telah meluangkan waktu, bersedia dan bersabar dalam membimbing, mengkritik, dan memberikan saran selama proses penulisan tugas akhir ini.
3. Bapak Seniman S.Kom., M.Kom. sebagai Dosen Penguji 1 dan Bapak Dr. Niskarto Zendrato S.Kom., M.Kom sebagai Dosen Penguji 2 yang telah memberikan kritik dan saran yang membangun selama penyusunan tugas akhir ini.
4. Ibu Dr. Maya Silvi Lydia B.Sc., M.Sc. selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
5. Bapak Dedy Arisandi S.T., M.kom., selaku Ketua Program Studi S1 Teknologi Informasi Universitas Sumatera Utara dan Bapak Ivan Jaya S.Si., M.Kom., selaku Sekretaris Program Studi S1 Teknologi Informasi Universitas Sumatera Utara.
6. Ibu Sarah Purnamawati S.T., M.Sc. selaku dosen Penasehat Akademik penulis
7. Seluruh jajaran Dosen dan *Staff* Program Studi S1 Teknologi Informasi yang telah menyampaikan ilmu yang bermanfaat dan membantu urusan administrasi penulis selama menjalani perkuliahan.

8. *Inang tua* sibatu batu, *Atturang* Vania, Vania, Vanesha, dan keluarga lain yang tidak dapat disebutkan yang telah membantu baik dalam dukungan moral maupun moril kepada penulis.
9. Kepada pemilik NIM 200803079, yang telah membantu dan menemani penulis dalam pengerjaan tugas akhir ini.
10. Teman-teman seperjuangan kuliah dari BIM dan JEMUT SQUAD yang telah menemani perjalanan perkuliahan penulis.
11. Teman-teman diskusi penulisan selama perkuliahan, yaitu Monika, Ruth dan Putri.
12. Semua teman-teman Kom A Stambuk 2020 Teknologi Informasi.

Penulis menyadari bahwa masih terdapat kekurangan dalam penyusunan tugas akhir ini. Oleh karena itu, penulis mengharapkan saran serta kritik yang membangun demi kesempurnaan tugas akhir ini. Akhir kata, semoga tugas akhir ini mampu memberikan pengetahuan dan informasi bagi pihak-pihak yang berkepentingan, baik akademis maupun non akademis.

Medan, 6 Maret 2025

Riski Hartanto Saragih

ABSTRAK

Penelitian ini bertujuan untuk mengembangkan sistem deteksi dini serangan jaringan dengan menggabungkan algoritma *Local Outlier Factor* (LOF) dan *Rule-based System*. LOF digunakan untuk mendeteksi anomali berdasarkan kepadatan data lokal, memungkinkan identifikasi *outlier* dengan akurasi tinggi pada data jaringan yang dinamis dan tidak berlabel. Algoritma ini mampu mengenali pola serangan yang sulit dideteksi oleh metode konvensional. Sementara itu, *Rule-based System* melengkapi LOF dengan pendekatan berbasis aturan "*if-then*" untuk mengenali pola serangan tertentu, meningkatkan akurasi dan validasi hasil deteksi. Kombinasi kedua metode ini bertujuan untuk meminimalkan *false positives* dan *false negatives*, meningkatkan respons terhadap serangan, dan memungkinkan tindakan mitigasi secara otomatis. Sistem yang dikembangkan diuji menggunakan data serangan jaringan yang meliputi DoS, *Probing*, TCP *Flood*, dan UDP *Flood*, yang diambil dari *dataset* publik yang tersedia. Proses pengujian melibatkan analisis keakuratan sistem dalam mendeteksi dan mengklasifikasikan serangan. Hasil penelitian menunjukkan bahwa integrasi LOF dan *Rule-based System* dapat mendeteksi berbagai jenis serangan dengan tingkat akurasi yang baik, meminimalkan kesalahan deteksi, dan memberikan informasi lalu lintas jaringan secara *real-time* kepada pengguna. Meskipun demikian, sistem ini masih bergantung pada aturan yang ditetapkan sebelumnya, yang berarti kemampuan deteksi terhadap pola serangan baru yang tidak teridentifikasi mungkin terbatas. Sistem ini juga belum dilengkapi dengan mekanisme otomatis untuk mencegah serangan setelah terdeteksi, sehingga hanya berfungsi sebagai alat pemantau dan deteksi dini. Oleh karena itu, penelitian ini menyarankan pengembangan lebih lanjut dengan mengintegrasikan teknologi pembelajaran mesin yang lebih adaptif seperti *deep learning*, serta penerapan mekanisme pencegahan otomatis untuk meningkatkan efektivitas deteksi dan respons terhadap ancaman jaringan.

Kata kunci: Anomali jaringan, Jaringan, *Local Outlier Factor* (LOF), *Rule-based System*

IDENTIFICATION OF NETWORK ATTACK ANOMALIES THROUGH A COMBINATION OF LOCAL OUTLIER FACTOR (LOF) AND RULE-BASED SYSTEM

ABSTRACT

This research aims to develop a network attack early detection system by combining the Local Outlier Factor (LOF) algorithm and Rule-Based System. LOF is used to detect anomalies based on local data density, enabling high-accuracy identification of outliers in dynamic and unlabeled network data. This algorithm is able to recognize attack patterns that are difficult to detect by conventional methods. Meanwhile, the Rule-based System complements LOF with an "if-then" rule-based approach to recognize specific attack patterns, improving the accuracy and validation of detection results. The combination of these two methods aims to minimize false positives and false negatives, improve response to attacks, and enable automatic mitigation actions. The developed system was tested using network attack data which includes DoS, Probing, TCP Flood, and UDP Flood, taken from publicly available datasets. The testing process involves analyzing the accuracy of the system in detecting and classifying attacks. The results show that the integration of LOF and Rule-Based System can detect various types of attacks with good accuracy, minimize detection errors, and provide real-time network traffic information to users. However, the system still relies on pre-defined rules, which means that the detection capability of new, unidentified attack patterns may be limited. The system is also not equipped with an automatic mechanism to prevent attacks once detected, so it only serves as a monitoring and early detection tool. Therefore, this research suggests further development by integrating more adaptive machine learning technologies such as deep learning, as well as the implementation of automated prevention mechanisms to improve the effectiveness of detection and response to network threats.

***Keywords**: Local Outlier Factor (LOF), Network, Network anomalies, Rule-based System*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Pada era ini, transisi digital memberikan sebuah lonjakan besar bagi peradaban umat manusia. Transisi digital merupakan perubahan yang berhubungan dengan penerapan teknologi digital dalam semua aspek kehidupan [1]. Perubahan tersebut memberikan banyak dampak positif seperti kemudahan akses informasi [2], inovasi teknologi baru [3], peningkatan efesiensi bisnis [3], kemajuan pendidikan [4] hingga pengingkatan akses kesehatan [5]. Meskipun demikian, transisi digital juga memberikan ancaman baru, khususnya pada bidang keamanan dan privasi [6]. Kebocoran data dan serangan *cyber* menjadi salah satu dampak serius yang perlu diperhatikan [7].

Di sisi lain, serangan *cyber* meningkat secara signifikan. Serangan-serangan tersebut berpotensi menyebabkan kebocoran data yang dapat menimbulkan kerugian yang sangat besar. Dengan demikian, hal ini harus diatasi secara serius dan menyeluruh untuk memastikan bahwa keamanan dan privasi pengguna tetap terjaga. Melonjaknya ancaman siber merupakan bayaran atas efisiensi yang telah didapatkan dari transisi digital. Beberapa diantaranya adalah DoS, *Probing*, U2R, dan R2l [8].

Serangan *Denial of Service* (DoS) adalah serangan siber yang bertujuan mengganggu fungsi normal suatu jaringan, sistem, atau layanan dengan membanjirinya dengan lalu lintas tidak sah, sehingga tidak tersedia bagi pengguna yang dituju [9]. Penyerang sering kali menggunakan aliran lalu lintas data berkecepatan tinggi dalam pola gelombang persegi untuk melakukan serangan DoS [10]. Aliran lalu lintas jaringan ini diulangi pada frekuensi tetap dalam skala waktu tertentu untuk membebani tautan jaringan korban dan mengganggu layanan mereka.

Serangan-serangan ini mengeksploitasi celah keamanan dengan mengirimkan sejumlah lalu lintas data yang tidak melebihi *bandwidth* korban, sehingga sulit untuk dideteksi dan dimitigasi.

Serangan *Probing* adalah jenis serangan di mana penyerang mencoba mengumpulkan data dan mengidentifikasi kerentanan dalam jaringan. Serangan *Probing* dilakukan oleh penyerang dengan mencari kelemahan dalam jaringan atau sistem [11]. Penyerang dapat menggunakan alat seperti nmap, satan, atau mscan untuk mengumpulkan informasi seperti alamat IP, nama layanan, sistem operasi, dan nama host, yang kemudian dapat digunakan sebagai dasar untuk serangan lebih lanjut. Serangan ini merupakan lanjutan dari serangan seperti DoS maupun serangan peretasan lainnya [12].

Serangan TCP *Flood Attack* merupakan jenis serangan *Denial of Service* (DoS) di mana seorang penyerang mengirimkan sejumlah besar paket TCP ke server target, yang mengakibatkan penggunaan sumber daya server yang berlebihan dan mencegahnya merespons permintaan yang sah [13].

Serangan UDP (*User Datagram Protocol*) *Flood Attack* juga merupakan jenis serangan *Denial of Service* (DoS) yang memanfaatkan protokol *User Datagram Protocol* (UDP) dengan mengirimkan sejumlah besar paket UDP ke *port* acak pada server target. Hal ini menyebabkan server berusaha merespons setiap paket, yang mengakibatkan penggunaan sumber daya CPU dan *bandwidth* yang sangat tinggi, sehingga layanan menjadi lambat atau tidak responsif [14].

Sebelumnya, T. Sui, et al. pada tahun 2020 telah melakukan penelitian yang membahas tentang penggunaan teori matriks acak dalam berbagai bidang seperti keuangan, biologi, jaringan pintar, komunikasi nirkabel, sistem tenaga, dan analisis data. Metodologi yang digunakan adalah DC-RMT yang baru untuk deteksi anomali dalam jaringan nirkabel menggunakan analisis *big* data. Metode ini efektif dalam mendeteksi anomali dalam data lalu lintas jaringan nirkabel yang sangat berkorelasi dan multidimensional.

Dengan menggunakan *Call Detail Records* (CDRs) yang dikumpulkan dari jaringan LTE nyata Telecom Italia di Milan penelitian ini menunjukkan bahwa anomali dapat berdampak signifikan pada kinerja jaringan dan bahwa deteksi anomali secara *real-time* sangat penting untuk optimasi jaringan. Metode yang diusulkan melebihi pendekatan berbasis pengelompokan dan dapat digunakan untuk memprediksi pola lalu

lintas jaringan. Dengan demikian, deteksi anomali seperti DoS yang akurat untuk alokasi sumber daya yang efisien dalam jaringan nirkabel sangat penting, terutama di era 5G dan seterusnya yang akan datang [15].

B. Min, et al. pada tahun 2021 melakukan penelitian mengenai metode deteksi anomali jaringan menggunakan *Memory-Augmented Deep Auto-Encoder* (MemAE) untuk meningkatkan deteksi perilaku jaringan yang abnormal. Model MemAE dilatih pada data normal dan menggunakan modul memori untuk merekonstruksi *input* yang abnormal lebih dekat dengan sampel normal, meningkatkan skor anomali untuk serangan.

Penelitian ini membahas beberapa jenis serangan yang diuji pada *dataset* NSL-KDD, UNSW-NB15, dan CICIDS 2017. Beberapa jenis serangan tersebut meliputi Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, dan Worms. Penelitian ini menunjukkan bahwa MemAE lebih unggul daripada model tradisional seperti *Autoencoder* dan *One-Class* SVM dalam mendeteksi anomali jaringan. *Encoding* langka dalam MemAE juga terbukti efektif dalam meningkatkan kinerja deteksi [16].

V. Rios, et al. pada tahun 2022 juga melakukan penelitian dengan tujuan untuk memberikan tinjauan komprehensif tentang serangan *Denial-of-Service* (DoS) dengan *Low-Rate Denial-of-Service* (LDoS), termasuk mekanisme deteksi dan pertahanan serta alat yang digunakan baik untuk serangan maupun pertahanan. Metode yang digunakan dalam penelitian ini melibatkan analisis berbagai studi penelitian dan pendekatan untuk mengurangi serangan LDoS, serta menyoroti pentingnya mengatasi serangan *cyber* yang bersifat menyelinap dan merusak. *Dataset* yang digunakan dalam penelitian ini berasal dari dua sumber, yaitu dari jaringan simulasi yang dihasilkan dan dari *dataset* CIC DoS yang tersedia secara publik. Di mana kedua *dataset* tersebut mencakup berbagai jenis serangan DoS dengan tingkat rendah, termasuk serangan *Reduction of Quality* (RoQ). Hasil yang diperoleh dari penelitian ini mencakup pengetahuan terkini tentang serangan LDoS bagi para peneliti dan administrator jaringan, serta memberikan panduan yang komprehensif tentang karakteristik serangan LDoS dan mekanisme pertahanannya [10].

S. Kamamura, et al. pada tahun 2023 melakukan penelitian mengenai metode deteksi anomali pada jaringan IP skala besar. Metode ini memanfaatkan serangkaian data lalu lintas waktu yang dibuat dari tanggal 1 April 2022 hingga 30 April 2022

dengan menggunakan generator lalu lintas. Data seri waktu dihasilkan dengan menyusun 10 aliran komunikasi yang berbeda dengan tingkat lalu lintas yang hampir sama. Penggunaan alat yang disebut Fast xFlow Proxy memungkinkan pemecahan aliran lalu lintas menjadi aliran individu dengan *granularity* yang baik.

Metode tersebut kemudian melakukan analisis korelasi sederhana dan konfigurasi ambang dinamis untuk mendeteksi anomali seperti serangan DDoS (*Distributed Denial of Service*) dan penurunan lalu lintas layanan akibat kegagalan layanan internet dalam aliran yang dipisahkan tersebut. Hasil simulasi menunjukkan tingkat akurasi deteksi anomali yang tinggi, bahkan dalam situasi sulit seperti fluktuasi lalu lintas kecil atau situasi berisik [17].

H. Kye, et al. pada tahun 2022 melakukan penelitian untuk membahas sistem deteksi intrusi jaringan yang memungkinkan deteksi hierarkis berbasis pembelajaran mandiri. Solusi yang diusulkan terdiri dari beberapa tahap deteksi, termasuk deteksi dini *outlier* ekstrim yang dapat menyebabkan kerusakan parah pada sistem. Selain itu, sistem ini melakukan pemeriksaan ulang menyeluruh menggunakan ruang tersembunyi dengan skor anomali khusus, yang menghasilkan tingkat akurasi deteksi yang tinggi.

Dalam penelitian ini, digunakan dua *dataset* yang populer yang dikumpulkan di jaringan LAN, yaitu *dataset* NSL-KDD dan *dataset* CSE-CIC-IDS2018. Dengan melakukan pengujian untuk melihat serangan DoS (*Denial of Service*), U2R (*User to Root*), R2L (*Remote to Local*), dan *Probing* dikonfirmasi bahwa solusi yang diusulkan dapat mendeteksi 20% data abnormal secara proaktif, dan dapat mendeteksi 99% data abnormal pada tahap akhir [18].

Berdasarkan penelitian terdahulu, identifikasi terhadap serangan siber seperti DoS (*Denial of Service*), U2R (*User to Root*), R2L (*Remote to Local*), dan *Probing* telah mendapatkan banyak perhatian dalam penelitian dan pengujian. Namun, tidak diberikan solusi yang efektif untuk dapat menyelesaikan dampak yang diberikan dari serangan-serangan tersebut. Disisi lain, serangan seperti TCP *Flood Attack*, UDP *Flood Attack* dan ICMP (Ping) *Flood Attack* merupakan ancaman paling umum yang berkembang. Penelitian ini menggunakan algoritma *Local Outlier Factor* (LOF) dan *Rule-based System* karena keduanya menawarkan pendekatan yang komplementer dalam mengidentifikasi anomali dan membuat keputusan cepat terhadap serangan jaringan. *Local Outlier Factor* (LOF) dipilih karena kemampuannya dalam mengidentifikasi anomali secara lokal berdasarkan kepadatan data di sekitarnya, yang mengidentifikasi

deteksi *outlier* dengan akurasi tinggi bahkan dalam data yang tidak berlabel atau berdistribusi tidak diketahui. Hal ini membuat *Local Outlier Factor* (LOF) sangat efektif dalam lingkungan jaringan yang dinamis dan kompleks, di mana pola serangan dapat bervariasi secara signifikan. Di sisi lain, *Rule-based System* digunakan untuk meningkatkan akurasi identifikasi dan validasi hasil dari *Local Outlier Factor* (LOF) dengan pendekatan yang lebih terstruktur melalui aturan "*if-then*" yang dirancang untuk mengenali pola serangan tertentu. Kombinasi kedua metode ini bertujuan untuk meminimalkan *false positives* dan *false negatives*, meningkatkan respons terhadap serangan, dan memungkinkan tindakan mitigasi secara otomatis. Integrasi *Local Outlier Factor (LOF)* dengan *Rule-based System* menyediakan solusi yang lebih *robust* dan adaptif dalam mengidentifikasi serangan jaringan secara dini dan mengambil tindakan yang diperlukan untuk mencegah dampak negatif lebih lanjut. Oleh karena itu, peneliti mengusulkan untuk melakukan penelitian tentang **Identifikasi Anomali Lalu Lintas Jaringan melalui Kombinasi *Local Outlier Factor* (LOF) dan *Rule-based System*** untuk meningkatkan kemampuan dalam mendeteksi serangan jaringan secara dini dan mengurangi dampak negatif yang mungkin timbul.

1.2. Rumusan Masalah

Identifikasi serangan jaringan adalah tantangan besar dalam melindungi infrastruktur dan data sensitif dari serangan siber yang terus berkembang dan semakin kompleks. Penyerang kini mampu menembus sistem dengan cara yang canggih dan sulit terdeteksi. Metode deteksi yang ada sering kali menghasilkan tingkat kesalahan tinggi, baik berupa deteksi yang salah (*false positives*) maupun serangan yang terlewat (*false negatives*), sehingga sulit membedakan aktivitas normal dari ancaman sebenarnya. Akibatnya, respons terhadap serangan menjadi lambat dan risiko keamanan meningkat. Untuk itu, pengembangan metode identifikasi menggunakan algoritma *Local Outlier Factor* (LOF) dan *Rule-based System* sangat penting untuk meningkatkan kemampuan identifikasi dini, sehingga langkah pencegahan dapat dilakukan lebih efektif guna mengurangi dampak serangan.

1.3. Tujuan Penelitian

Penelitian ini bertujuan untuk membuat sebuah sistem pengidentifikasi anomali lalu lintas jaringan yang menyerang sebuah sistem jaringan dan berperan untuk melindungi sistem jaringan dari serangan jaringan.

1.4. Batasan Masalah Penelitian

Batasan masalah pada penelitian ini antara lain :

1. Sistem yang dibangun hanya mencakup parameter yang digunakan, yaitu DoS, *Probing*, TCP *Flood* dan UDP *Flood*.
2. Sistem hanya dapat mengidentifikasi namun tidak dapat menghentikan serangan dari luar.
3. Pengujian program berupa simulasi dengan *dataset*.
4. Sistem dibangun berdasarkan fungsi identifikasi sehingga tidak dapat memberikan perlindungan nyata terhadap sistem jaringan.
5. Sistem dibangun untuk sistem jaringan personal.
6. *Output* sistem yang dibangun merupakan tampilan grafik dari algoritma *Local Outliner Factor* (LOF) dengan informasi lalu lintas jaringan.

1.5. Manfaat Penelitian

Manfaat yang diperoleh dari penelitian yang dilakukan antara lain :

1. Pengurangan *False Positive* dan *False Negatif* dengan meningkatkan akurasi identifikasi.
2. Mempermudah pengidentifikasian serangan jaringan pada sistem jaringan.
3. Melindungi sistem jaringan dari serangan jaringan sehingga mengurangi kerugian masif dari serangan tersebut.
4. Optimasi sumber daya dengan pengidentifikasian yang lebih akurat akan membantu profesional keamanan jaringan dalam mengambil tindakan responsif dan efektif terhadap serangan jaringan.

1.6. Metodologi Penelitian

1. Studi literatur, diperlukanya studi literatur adalah bentuk upaya pencarian informasi yang dibutuhkan untuk menjadi referensi penelitian, referensi penelitian dapat berasal dari buku, *e-book*, jurnal, artikel ilmiah, dan *website* terpercaya.
2. Identifikasi masalah yang bertujuan untuk menemukan hal yang menjadi permasalahan, sehingga dapat disimpulkan rumusan masalah dan batasan masalah agar fokus dari penelitian ini terarah dan menemukan solusi yang jelas dan tepat.
3. Perancangan program. Tahap ini dilakukan untuk membuat *blueprint* dari program agar sesuai dengan kebutuhan dan solusi dari permasalahan yang ada.
4. Implementasi program, pada tahap ini rancangan program yang sebelumnya dibuat akan diimplementasikan ke dalam bentuk program dan siap untuk uji coba.
5. Pengujian sistem, tahap ini dilakukan untuk menguji fungsi dan keefektifan program yang dibuat, hasil yang diperoleh akan dikumpulkan dan dibawa untuk dibahas di laporan akhir penelitian.
6. Penyusunan laporan sebagai tahap akhir dari penelitian, pada tahap ini seluruh dokumentasi dari proses penelitian akan dikumpulkan lalu disusun menjadi sebuah skripsi utuh.

BAB 2

LANDASAN TEORI

2.1. *Local Outlier Factor* (LOF)

Local Outlier Factor (LOF) adalah algoritma yang digunakan untuk mendeteksi anomali atau *outlier* dalam suatu *dataset*. Algoritma ini pertama kali diusulkan oleh Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, dan Jörg Sander dalam makalah mereka pada tahun 2000. LOF bekerja dengan pendekatan berbasis kepadatan, di mana algoritma ini mengevaluasi kepadatan lokal dari suatu titik data dan membandingkannya dengan kepadatan titik-titik data di sekitarnya. Untuk melakukan ini, LOF menghitung jarak ke k tetangga terdekat (k-*nearest neighbors*, kNN) untuk setiap titik dalam *dataset*. Kemudian, algoritma ini mengukur rasio kepadatan lokal (*Local Reachability Density*) dari titik target terhadap tetangga-tetangganya. Jika kepadatan lokal suatu titik jauh lebih rendah dibandingkan dengan tetangganya, maka titik tersebut dianggap sebagai *outlier*. Nilai LOF yang dihasilkan menunjukkan tingkat keanehan (*outlierness*) suatu titik. Nilai LOF lebih besar dari 1 menunjukkan bahwa titik tersebut adalah *outlier*; semakin tinggi nilainya, semakin besar kemungkinannya menjadi *outlier*. LOF adalah metode deteksi *outlier* yang tidak diawasi (*unsupervised*), yang berarti dapat diterapkan pada data yang tidak diberi label, cocok untuk situasi di mana data dalam jumlah besar tidak dapat diberi label secara manual, seperti dalam jaringan komputer [19].

Selain dari informasi dasar mengenai LOF, algoritma ini telah digunakan secara luas dalam berbagai bidang seperti deteksi penipuan, pengawasan kesehatan, dan keamanan siber. Misalnya, dalam deteksi penipuan kartu kredit, LOF dapat membantu mengidentifikasi transaksi yang mencurigakan dengan membandingkan pola transaksi terbaru dengan pola transaksi yang normal. Dalam pengawasan kesehatan, LOF dapat digunakan untuk mendeteksi anomali dalam data pasien, seperti perubahan mendadak dalam tanda vital yang dapat menunjukkan kondisi medis yang memerlukan perhatian

segera. Keunggulan utama LOF adalah kemampuannya untuk mendeteksi anomali dalam data yang tidak memiliki distribusi yang diketahui atau yang tidak dapat diasumsikan memiliki distribusi tertentu, membuatnya fleksibel untuk berbagai jenis aplikasi [20].

2.2. *Rule-based System*

Sistem berbasis aturan atau "*Rule-based System"* adalah salah satu bentuk sistem kecerdasan buatan yang paling sederhana, yang menggunakan aturan-aturan sebagai representasi pengetahuan. Aturan-aturan ini sering kali berupa pernyataan "jika-maka" (*if-then*), yang menentukan tindakan yang harus diambil atau kesimpulan yang harus dicapai dalam situasi tertentu. Konsep ini berkembang seiring dengan perkembangan bidang kecerdasan buatan dan sistem pakar pada tahun 1970-an dan 1980-an, dengan kontribusi dari banyak peneliti, termasuk Edward Feigenbaum, yang dikenal sebagai pelopor dalam bidang sistem pakar [21].

Cara kerja sistem berbasis aturan melibatkan beberapa elemen dasar, yaitu kumpulan fakta yang relevan dengan kondisi awal, kumpulan aturan yang berisi tindakan yang harus diambil, dan kriteria terminasi untuk menentukan bahwa solusi telah ditemukan atau tidak ada solusi yang mungkin. Dalam aplikasinya, sistem ini terdiri dari basis pengetahuan yang menyimpan aturan-aturan, basis data yang terdiri dari fakta-fakta, mesin inferensi yang menjalankan proses penalaran, sub sistem penjelasan yang menganalisis dan menjelaskan penalaran kepada pengguna, serta antarmuka pengguna untuk komunikasi antara pengguna dan sistem.

Sistem berbasis aturan dapat bekerja dengan dua metode penalaran utama: penalaran maju (*forward chaining*) yang dimulai dengan fakta yang diketahui dan menerapkan aturan untuk menyimpulkan fakta baru, dan penalaran mundur (*backward chaining*) yang dimulai dengan tujuan dan bekerja mundur untuk menentukan fakta yang mendukung tujuan tersebut. Salah satu aplikasi terkenal dari sistem berbasis aturan adalah MYCIN, yang dikembangkan pada tahun 1970-an untuk diagnosis infeksi bakteri dan rekomendasi pengobatan berdasarkan aturan medis. Sistem ini menunjukkan bagaimana aturan-aturan dapat digunakan untuk membuat keputusan yang kompleks dan bermakna dalam berbagai bidang.

2.3. *Distributed Denial of Service* (DDoS) atau *Denial of Service* (DoS)

Serangan *Denial of Service* (DoS) dan *Distributed Denial of Service* (DDoS) adalah upaya yang dilakukan oleh pihak penyerang untuk membuat suatu layanan atau sumber daya jaringan tidak dapat diakses oleh pengguna yang sah. Serangan ini dilakukan dengan cara membanjiri target dengan sejumlah besar permintaan palsu atau data yang melebihi kapasitas penanganan dari sistem target, sehingga menyebabkan kegagalan layanan [22].

Serangan DoS dan DDoS dapat terjadi melalui berbagai vektor serangan seperti *exhaustion resources attacks* atau *volumetric attacks*, termasuk serangan amplifikasi/refleksi seperti Smurf, Fraggle, NTP amplification, dan DNS *amplification*. Dalam serangan *flooding*, tujuannya adalah untuk menghabiskan sumber daya kritis jaringan seperti *bandwidth* atau kapasitas pemrosesan dengan menggunakan metode seperti UDP *flooding*, ICMP *flooding*, *link flooding*, TCP SYN *flooding*, dan *ping of death*.

Serangan DoS pertama kali dikenali pada tahun 1996 oleh Eugene Kashpureff yang menyerang situs *web* Internet *Software Consortium* dengan serangan DNS *hijacking*. Serangan DDoS pertama kali mencuat di tahun 2000 ketika seorang remaja Kanada dengan nama samaran “Mafiaboy” melancarkan serangan yang melumpuhkan situs-situs besar seperti Yahoo!, Amazon, CNN, dan eBay, menyebabkan kerugian finansial yang signifikan dan menarik perhatian dunia terhadap bahaya serangan semacam itu.

Kasus terkenal lainnya termasuk serangan DDoS pada tahun 2016 terhadap Dyn, sebuah perusahaan penyedia layanan DNS, yang menyebabkan pemadaman internet di seluruh Amerika Serikat dan Eropa. Serangan ini dilakukan dengan menggunakan botnet Mirai, yang menginfeksi perangkat IoT (*Internet of Things*) dan menggunakannya untuk mengirimkan lalu lintas yang sangat besar ke target.

Diluar itu, serangan DDoS terus berkembang dengan taktik yang semakin canggih, termasuk penggunaan botnet yang lebih besar dan lebih terorganisir. Upaya mitigasi mencakup penggunaan teknologi seperti *cloud-based* DDoS *protection services*, AI dan *machine learning* untuk mendeteksi pola serangan, serta strategi mitigasi berbasis *blockchain* yang menawarkan pendekatan desentralisasi untuk meningkatkan keamanan jaringan.

2.4. *Probing*

Serangan *Probing* adalah jenis serangan siber yang bertujuan untuk mengumpulkan informasi tentang target, seperti sistem operasi, layanan yang berjalan, dan kerentanan yang ada. Informasi ini kemudian dapat digunakan oleh penyerang untuk merencanakan serangan yang lebih terarah. Serangan *Probing* biasanya dilakukan pada tahap awal serangan, yang dikenal sebagai fase pengintaian dalam rantai pembunuhan siber. Serangan ini dapat dilakukan dengan berbagai cara, seperti pemindaian *port* (*port scanning*), *ping sweep*, dan *vulnerability scanning* [12].

Port scanning melibatkan pengiriman paket ke berbagai *port* pada sistem target untuk melihat *port* mana yang terbuka dan layanan apa yang berjalan pada *port* tersebut. *Ping sweep* melibatkan pengiriman permintaan *ping* ke berbagai alamat IP untuk melihat *host* mana yang aktif. *Vulnerability scanning* melibatkan penggunaan alat otomatis untuk memindai sistem target untuk mencari kerentanan yang diketahui.

Dilansir dari *cyberlaw.ccdcoe.org* serangan *Probing* yang terkenal adalah serangan terhadap Sony Pictures Entertainment pada tanggal 24 November 2014. Dalam serangan ini, para penyerang menggunakan serangan *Probing* untuk mengumpulkan informasi tentang jaringan Sony sebelum meluncurkan serangan yang lebih merusak yang mengakibatkan kebocoran data sensitif. Serangan *Probing* dapat menjadi ancaman serius bagi keamanan jaringan. Dengan memahami cara kerja dan dampaknya, organisasi dapat mengambil langkah-langkah untuk melindungi sistem mereka dari serangan ini.

2.5. TCP *Flood Attack*

Sejak ditemukan, serangan TCP *Flood*, khususnya serangan *flood* TCP SYN, telah menjadi perhatian signifikan dalam keamanan jaringan. Serangan *Flood* TCP SYN mengeksploitasi mekanisme *three-way handshake* TCP, yang penting untuk membangun koneksi antara klien dan server. Selama *handshake* ini, klien mengirimkan paket SYN (*synchronize*), server merespons dengan paket SYN-ACK (*synchronize-acknowledge*), dan klien menyelesaikan koneksi dengan paket ACK (*acknowledge*). Dalam serangan SYN *flood*, penyerang mengirimkan SYN *packet* dengan jumlah banyak dari alamat IP palsu ke server target, sehingga menyebabkan server mengalokasikan sumber daya untuk koneksi yang tidak pernah diselesaikan. Hal ini

mengarah pada kehabisan sumber daya dan gangguan layanan bagi pengguna yang sah [13].

Kasus yang cukup dikenal dari serangan *flood* TCP SYN terjadi di tahun 2000 pada Yahoo!, yang merupakan salah satu kasus profil tinggi pertama. Serangan ini membanjiri server Yahoo! dengan permintaan koneksi palsu, membuat situs tersebut tidak dapat diakses selama beberapa jam. Peristiwa besar lainnya terjadi pada tahun 2016, yang menargetkan penyedia DNS Dyn. Serangan ini, sebagai bagian dari serangan *Distributed Denial of Service* (DDoS) yang lebih besar, melibatkan beberapa vektor serangan, termasuk *flood* TCP SYN, dan menyebabkan gangguan internet yang meluas.

Seiring dengan berjalannya waktu, deteksi dan pencegahan serangan *flood* TCP SYN semakin berkembang. Teknik modern, seperti penyesuaian ambang batas adaptif, secara dinamis menyesuaikan ambang batas untuk tingkat paket SYN normal berdasarkan pola lalu lintas terbaru. Ketika jumlah paket SYN melebihi ambang batas adaptif ini, sistem mengklasifikasikan lalu lintas sebagai berbahaya dan dapat mengambil tindakan seperti memblokir alamat IP yang terlibat.

2.6. UDP *Flood Attack*

UDP (*User Datagram Protocol*) *Flood Attack* adalah jenis serangan yang membanjiri jaringan dengan UDP *packets*. Dalam serangan ini, sistem jahat atau bot mengirimkan sejumlah besar datagram UDP secara bersamaan dengan ukuran yang bervariasi. Tujuan utamanya adalah menyebabkan kemacetan pada jaringan dan *Denial of Service* (DoS) pada sistem target, membuat layanan dan sumber daya tidak dapat diakses oleh pengguna yang sah. Serangan UDP *flood* telah dikenal sejak protokol UDP diperkenalkan dan mulai digunakan dalam jaringan komputer. Tidak ada tanggal pasti kapan serangan pertama kali ditemukan, namun serangan DoS, termasuk UDP *flood*, telah menjadi perhatian sejak internet mulai berkembang pesat pada tahun 1990-an.

Menurut Liputan6 pada tanggal 25 Desember 2014, serangan UDP *flood* yang terkenal adalah serangan pada layanan Xbox Live dan PlayStation Network pada tahun 2014, di mana kedua *platform game online* ini mengalami gangguan besar akibat serangan DDoS yang melibatkan UDP *flood* yang menyebabkan *downtime* yang signifikan bagi pengguna di seluruh dunia. Selain itu, dilansir dari TheGuardian pada tanggal 26 Oktober 2016 serangan pada Dyn DNS yang memanfaatkan Mirai botnet,

juga menggunakan teknik UDP *flood*. Serangan ini menyebabkan banyak situs *web* populer, termasuk Twitter, Reddit, dan Spotify, menjadi tidak dapat diakses untuk sementara waktu.

Eksperimen yang dilakukan oleh A. Bijalwan, et al. menunjukkan bahwa bot mengirimkan datagram UDP berukuran berbeda seperti 126, 124, dan 120 *byte* pada 22 Mei 2014. Total paket yang dipertukarkan selama serangan adalah 8895, dibandingkan dengan hanya 192 paket dalam aliran normal. UDP *Flood Attack* adalah ancaman signifikan terhadap ketersediaan layanan jaringan. Dengan mengirimkan sejumlah besar datagram UDP, penyerang dapat dengan mudah melumpuhkan sistem target, membuatnya tidak dapat melayani pengguna yang sah. Pemahaman dan deteksi dini terhadap pola serangan ini sangat penting untuk melindungi infrastruktur jaringan dari serangan yang merusak [14].

2.7. Penelitian Terdahulu

Hingga saat ini, telah banyak dilakukan penelitian mengenai serangan jaringan seperti DoS (*Denial of Service*), U2R (*User to Root*), R2L (*Remote to Local*). Berbagai metode juga telah banyak digunakan untuk mendeteksi anomali dalam jaringan nirkabel. Tabel 2.1 memuat beberapa penelitian terdahulu yang membahas metode deteksi serangan jaringan beserta jenis serangan jaringannya.

Tabel 2.1 Penelitian Terdahulu

No.	Penulis/Tahun	Metode	Keterangan
1.	Tengfei Sui, Xiaofeng Tao, Shida Xia, Hui Chen, Huici Wu, Xuefei Zhang, dan Kechen Chen (2020)	DC-RMT	Metode yang diusulkan melebihi pendekatan berbasis pengelompokan dan dapat digunakan untuk memprediksi pola lalu lintas jaringan. Dengan demikian, deteksi anomali seperti DoS yang akurat untuk alokasi sumber daya yang efisien dalam jaringan nirkabel sangat penting, terutama di era 5G dan seterusnya yang akan datang.

No.	Penulis/Tahun	Metode	Keterangan
2.	Byeongjun Min, Jihoon Yoo, Sangsoo Kim, Dongil Shin, dan Dongkyoo Shin (2021)	*Memory-Augmented Deep Auto-Encoder* (MemAE)	Melalui penelitian ini diperoleh bahwa MemAE lebih unggul daripada model tradisional seperti *Autoencoder* dan *One-Class* SVM dalam mendeteksi anomali jaringan. *Encoding* langka dalam MemAE juga terbukti efektif dalam meningkatkan kinerja deteksi.
3.	Vinícius De Miranda Rios, Pedro R. M. Inácio, Damien Magoni, dan Mário M. Freire (2022)	*Analysis and Filtering Module* (AFM), *Chi-square Test*, *Machine Learning Algorithms*, *Genetic Algorithm*, *Fuzzy Logic*, *Neural Network*, dan *Euclidean Distance dan Adaptive* KPCA (*Kernel Principal Component Analysis*)	Hasil yang diperoleh dari penelitian ini mencakup pengetahuan terkini tentang serangan LDoS bagi para peneliti dan administrator jaringan, serta memberikan panduan yang komprehensif tentang karakteristik serangan LDoS dan mekanisme pertahanannya
4.	Shohei Kamamura, Yuki Takei, Masato Nishiguchi, Yuhei Hayashi, dan Takayuki Fujiwara (2023)	Analysis of Correlation terhadap pola *time-series*	Hasil simulasi yang dilakukan menunjukkan tingkat akurasi deteksi anomali yang tinggi, bahkan dalam situasi sulit seperti fluktuasi lalu lintas kecil atau situasi berisik.
5.	Hyoseon Kye, Miru Kim, dan	*Hierarchical Detection* berbasis *A*	Dengan melakukan pemeriksaan ulang secara menyeluruh menggunakan ruang tersembunyi

No.	Penulis/Tahun	Metode	Keterangan
	Minhae Kwon (2022)	*Self-Supervised Learning*	dengan skor anomali khusus, yang menghasilkan tingkat akurasi deteksi yang tinggi, dikonfirmasi bahwa solusi yang diusulkan dapat mendeteksi 20% data abnormal secara proaktif, dan dapat mendeteksi 99% data abnormal pada tahap akhir.

T. Sui, et al. pada tahun 2020 telah melakukan penelitian yang membahas tentang penggunaan teori matriks acak dalam berbagai bidang seperti keuangan, biologi, jaringan pintar, komunikasi nirkabel, sistem tenaga, dan analisis data. Metodologi yang digunakan adalah DC-RMT yang baru untuk deteksi anomali dalam jaringan nirkabel menggunakan analisis *big* data. Metode ini efektif dalam mendeteksi anomali dalam data lalu lintas jaringan nirkabel yang sangat berkorelasi dan multidimensional.

Dengan menggunakan *Call Detail Records* (CDRs) yang dikumpulkan dari jaringan LTE nyata Telecom Italia di Milan penelitian ini menunjukkan bahwa anomali dapat berdampak signifikan pada kinerja jaringan dan bahwa deteksi anomali secara *real-time* sangat penting untuk optimasi jaringan. Metode yang diusulkan melebihi pendekatan berbasis pengelompokan dan dapat digunakan untuk memprediksi pola lalu lintas jaringan. Dengan demikian, deteksi anomali seperti DoS yang akurat untuk alokasi sumber daya yang efisien dalam jaringan nirkabel sangat penting, terutama di era 5G dan seterusnya yang akan datang [15].

B. Min, et al. pada tahun 2021 melakukan penelitian mengenai metode deteksi anomali jaringan menggunakan *Memory-Augmented Deep Auto-Encoder* (MemAE) untuk meningkatkan deteksi perilaku jaringan yang abnormal. Model MemAE dilatih pada data normal dan menggunakan modul memori untuk merekonstruksi *input* yang abnormal lebih dekat dengan sampel normal, meningkatkan skor anomali untuk serangan.

Penelitian ini membahas beberapa jenis serangan yang diuji pada *dataset* NSL-KDD, UNSW-NB15, dan CICIDS 2017. Beberapa jenis serangan tersebut meliputi

Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, dan Worms. Penelitian ini menunjukkan bahwa MemAE lebih unggul daripada model tradisional seperti *Autoencoder* dan *One-Class* SVM dalam mendeteksi anomali jaringan. *Encoding* langka dalam MemAE juga terbukti efektif dalam meningkatkan kinerja deteksi [16].

V. Rios, et al. pada tahun 2022 juga melakukan penelitian dengan tujuan untuk memberikan tinjauan komprehensif tentang serangan *Denial-of-Service* (DoS) dengan *Low-Rate Denial-of-Service* (LDoS), termasuk mekanisme deteksi dan pertahanan serta alat yang digunakan baik untuk serangan maupun pertahanan. Metode yang digunakan dalam penelitian ini melibatkan analisis berbagai studi penelitian dan pendekatan untuk mengurangi serangan LDoS, serta menyoroti pentingnya mengatasi serangan *cyber* yang bersifat menyelinap dan merusak. *Dataset* yang digunakan dalam penelitian ini berasal dari dua sumber, yaitu dari jaringan simulasi yang dihasilkan dan dari *dataset* CIC DoS yang tersedia secara publik. Di mana kedua *dataset* tersebut mencakup berbagai jenis serangan DoS dengan tingkat rendah, termasuk serangan *Reduction of Quality* (RoQ). Hasil yang diperoleh dari penelitian ini mencakup pengetahuan terkini tentang serangan LDoS bagi para peneliti dan administrator jaringan, serta memberikan panduan yang komprehensif tentang karakteristik serangan LDoS dan mekanisme pertahanannya [10].

S. Kamamura, et al. pada tahun 2023 melakukan penelitian mengenai metode deteksi anomali pada jaringan IP skala besar. Metode ini memanfaatkan serangkaian data lalu lintas waktu yang dibuat dari tanggal 1 April 2022 hingga 30 April 2022 dengan menggunakan generator lalu lintas. Data seri waktu dihasilkan dengan menyusun 10 aliran komunikasi yang berbeda dengan tingkat lalu lintas yang hampir sama. Penggunaan alat yang disebut Fast xFlow Proxy memungkinkan pemecahan aliran lalu lintas menjadi aliran individu dengan *granularity* yang baik.

Metode tersebut kemudian melakukan analisis korelasi sederhana dan konfigurasi ambang dinamis untuk mendeteksi anomali seperti serangan DDoS (*Distributed Denial of Service*) dan penurunan lalu lintas layanan akibat kegagalan layanan internet dalam aliran yang dipisahkan tersebut. Hasil simulasi menunjukkan tingkat akurasi deteksi anomali yang tinggi, bahkan dalam situasi sulit seperti fluktuasi lalu lintas kecil atau situasi berisik [17].

H. Kye, et al. pada tahun 2022 melakukan penelitian untuk membahas sistem deteksi intrusi jaringan yang memungkinkan deteksi hierarkis berbasis pembelajaran mandiri. Solusi yang diusulkan terdiri dari beberapa tahap deteksi, termasuk deteksi dini *outlier* ekstrim yang dapat menyebabkan kerusakan parah pada sistem. Selain itu, sistem ini melakukan pemeriksaan ulang menyeluruh menggunakan ruang tersembunyi dengan skor anomali khusus, yang menghasilkan tingkat akurasi deteksi yang tinggi.

Dalam penelitian ini, digunakan dua *dataset* yang populer yang dikumpulkan di jaringan LAN, yaitu *dataset* NSL-KDD dan *dataset* CSE-CIC-IDS2018. Dengan melakukan pengujian untuk melihat serangan DoS (*Denial of Service*), U2R (*User to Root*), R2L (*Remote to Local*), dan *Probing* dikonfirmasi bahwa solusi yang diusulkan dapat mendeteksi 20% data abnormal secara proaktif, dan dapat mendeteksi 99% data abnormal pada tahap akhir [18].

2.8. Perbedaan Penelitian

Penelitian yang ditulis ini dibuat dengan mengambil perbedaan dari penelitian yang sudah ada sebelumnya, yang mana pada penelitian T. sui et al. (2020) menggunakan teori matriks acak untuk mendeteksi anomali dalam jaringan nirkabel melalui analisis *big* data, sedangkan penelitian ini menggunakan *outliner* jaringan untuk mendeteksi anomali pada *dataset* yang telah tersedia di internet.

Dari penelitian B. Min, et al. (2021) yang menggunakan metode *deep-learning* model *Memory-Augmented Deep Auto-Encoder* (MemAE) yang memanfaatkan modul memori untuk merekonstruksi *input* yang tidak normal untuk mendeteksi anomali pada jaringan, namun pada penelitian ini tidak menggunakan *deep-learning* untuk implementasi yang lebih praktis dalam berbagai lingkungan jaringan.

Perbedaan juga terletak di tipe serangan yang digunakan, yang mana pada penelitian yang dilakukan oleh V. Rios, et al. (2022) yang menggunakan tipe serangan *low-rate Denial of Services* yang berintensitas rendah sedangkan pada penelitian ini menggunakan tipe serangan *Distributed Denial of Service* yang tergolong serangan berintensitas tinggi.

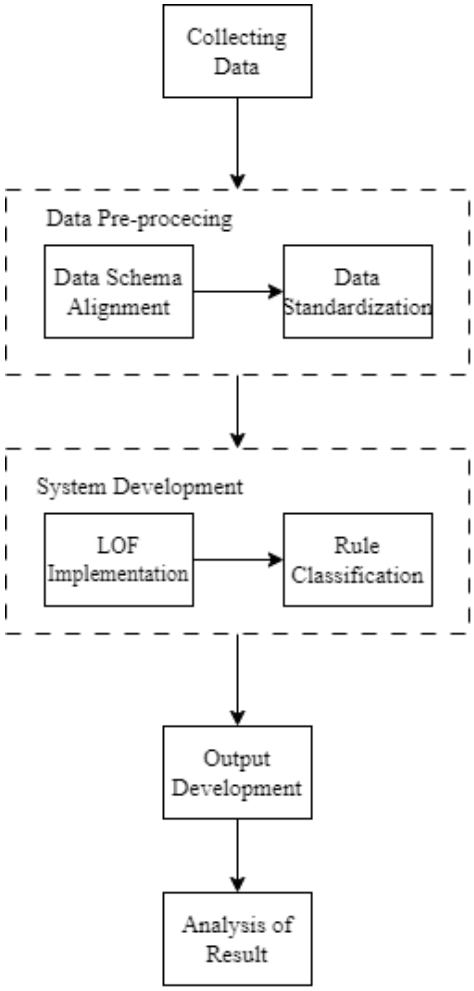
Selanjutnya, penelitian yang dilakukan oleh S. Kamamura, et al. (2023) menggunakan metode deteksi anomali pada jaringan dengan IP besar. Metode ini menggunakan serangkaian data lalu lintas Waktu yang dibuat dengan menggunakan generator lalu lintas dan menggunakan alat yang disebut *Fast xFlow Proxy* yang

memungkinkan pemecahan aliran lalu lintas menjadi aliran individu dengan *grunalarity* yang baik, sedangkan penelitian ini menggunakan alamat ip tertentu dan menggunakan simulasi komputer untuk pengaplikasiannya.

Dari penelitian yang dilakukan oleh Kye et al. (2022) membahas sistem intrusi jaringan yang memungkinkan deteksi hierarkis berbasis pembelajaran mandiri. Penelitian ini dapat mendeteksi *outlier* (anomali) ekstrim yang dapat menyebabkan kerusakan parah pada *system*, sedangkan penelitian ini hanya dapat mendeteksi anomali tingkat sedang hingga tinggi.

BAB 3

ANALISIS DAN PERANCANGAN SISTEM



Gambar 3.1 Arsitektur Umum

3.1. Data *Collection*

Data *collection* merupakan tahapan awal dari penelitian ini. Data serangan *cyber* diambil dari *dataset* kaggle https://www.kaggle.com/datasets/anushonkar/network-anamoly-detection/data oleh Anushokar sebagai *dataset* 1. *Dataset* tersebut merupakan *dataset* anomali serangan jaringan karena berisi data berbagai jenis serangan (anomali) serangan jaringan. *Dataset* tersebut digunakan untuk mendeteksi penyimpangan atau

serangan dalam jaringan komputer dengan menganalisis berbagai fitur koneksi seperti durasi, jumlah *byte* yang ditransfer, tingkat kesalahan, dan statistik lainnya. Setiap kategori serangan memiliki karakteristik dan pola tersendiri yang dapat diidentifikasi melalui fitur-fitur tertentu dalam *dataset* dan https://www.kaggle.com/datasets/amanverma1999/a-comprehensive-dataset-for-ddos-attack oleh Aman Verma sebagai *dataset* 2. Data tersebut berisi beberapa jenis serangan.

3.2. Pre-*processing*

Pada tahap ini format data yang telah dikumpulkan diubah menjadi .CSV dan perlu dilakukan *data schema alignment* dan *data Standardization*. Kedua *dataset* yang diambil memiliki nama kolom yang berbeda namun memiliki nilai yang sama sehingga dibutuhkan penyamaan skema agar kedua *dataset* tersebut dapat digunakan pada sistem yang akan dibangun.

Tabel 3.1 Cuplikan *Dataset* 1 dan *Dataset* 2 sebelum dilakukan *Schema Data Alignment*

protocoltype	service	srcbytes	dstbytes	count	srvcount
tcp	ftp_data	491	0	2	2
udp	other	146	0	13	1
tcp	private	0	0	123	6
tcp	http	232	8153	5	5
tcp	http	199	420	30	32
tcp	private	0	0	121	19
tcp	private	0	0	166	9
tcp	private	0	0	117	16
tcp	remote_job	0	0	270	23

udp	private	105	146	1	1
udp	private	105	146	1	1
udp	private	105	146	1	1
udp	private	105	146	2	2
udp	private	105	146	2	2
udp	private	105	146	2	2
udp	domain_u	29	0	2	1
udp	private	105	146	1	1
udp	private	105	146	2	2
tcp	http	223	185	4	4

Tabel 3.2 Cuplikan *Dataset* 1 dan *Dataset* 2 setelah dilakukan *Schema Data Alignment*

protocoltype	service	srcbytes	dstbytes	count	srvcount
tcp	ftp_data	491	0	2	2

udp	other	146	0	13	1
tcp	private	0	0	123	6
tcp	http	232	8153	5	5
tcp	http	199	420	30	32
tcp	private	0	0	121	19
tcp	private	0	0	166	9
tcp	private	0	0	117	16
tcp	remote_job	0	0	270	23

protocoltype	service	srcbytes	dstbytes	count	srvcount
udp	private	105	146	1	1
udp	private	105	146	1	1
udp	private	105	146	2	2
udp	private	105	146	2	2
udp	private	105	146	2	2
udp	domain_u	29	0	2	1
udp	private	105	146	1	1
udp	private	105	146	2	2
tcp	http	223	185	4	4

Data yang telah melawati proses penyamaan skema selanjutnya akan di standarisasi. Standarisasi ini berguna untuk menjaga kestabilan algoritma yang sensitif terhadap skala data dan meningkatkan performa model sistem yang dibangun. Dengan menggunakan *dataset* 1 dan *dataset* 2 yang telah melewati proses penyamaan skema seperti pada Tabel 3.2 dilakukan standarisasi dengan menggunakan skrip berikut.

```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(numerical_features)
```

Gambar 3.2 Standarisasi *Scale*

Fungsi StandardScaler dari *library scikit-learn* [23] digunakan untuk melakukan standardisasi (normalisasi z-*score*) pada data numerik. Standardisasi mengubah data sehingga memiliki rata-rata (*mean*) 0 dan standar deviasi 1. Hal ini dilakukan menggunakan rumus berikut:

$$z = \frac{x - \mu}{\sigma} \tag{3.1}$$

di mana:

$z =$ nilai hasil standarisasi

$x =$ nilai asli

$\mu =$ rata-rata (*mean*) dari fitur

$\sigma =$ standar deviasi dari fitur

Misalkan ada *dataset* numerik sederhana:

Fitur: $[10, 20, 30, 40, 50]$

1. Hitung rata-rata (μ) dan standar deviasi (σ)
   a. Rata-rata (μ)

$$\mu = \frac{10 + 20 + 30 + 40 + 50}{5} = 30$$

   b. Standar Deviasi

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{n}}$$

$$= \sqrt{\frac{(10-30)^2 + (20-30)^2 + (30-30)^2 + (40-30)^2 + (50-30)^2}{5}}$$

$$= \sqrt{\frac{400 + 100 + 0 + 100 + 400}{5}}$$

$$= \sqrt{200}$$

$$\approx 14.14$$

2. Hitung nilai Standardisasi (z) untuk setiap data
   a. Untuk $x = 10$

$$z = \frac{10 - 30}{14.14} = \frac{-20}{14.14} \approx -1.41$$

   b. Untuk $x = 20$

$$z = \frac{20 - 30}{14.14} = \frac{-10}{14.14} \approx -0.71$$

   c. Untuk $x = 30$

$$z = \frac{30 - 30}{14.14} = 0$$

   d. Untuk $x = 40$

$$z = \frac{40 - 30}{14.14} = \frac{10}{14.14} \approx 0.71$$

   e. Untuk $x = 50$

$$z = \frac{50 - 30}{14.14} = \frac{20}{14.14} \approx 1.41$$

3. Hasil data yang di standardisasi

Standardized Data: $[-1.41, -0.71, 0, 0.71, 1.41]$

Hasil standarisasi yang dilakukan pada *dataset* 1 dan *dataset* 2 akan tampak seperti pada Tabel 3.3 berikut.

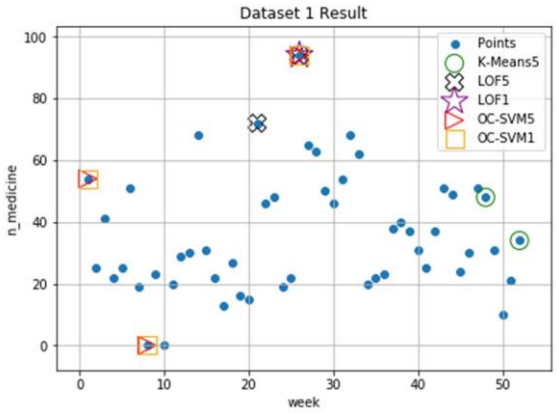
Tabel 3.3 Cuplikan *Dataset* 1 dan *Dataset* 2 setelah dilakukan Standarisasi

duration	srcbytes	dstbytes	land	wrongfragment
-0.155552356	-0.204374408	-0.370567887	0	0
-0.155552356	-0.204374408	-0.370567887	0	0
-0.155552356	-0.204374408	-0.370567887	0	0
-0.155552356	-0.204374408	-0.370567887	0	0
-0.155552356	-0.204374408	-0.370567887	0	0
-0.155552356	-0.223984608	-0.417554652	0	0
-0.155552356	-0.204374408	-0.370567887	0	0
-0.155552356	-0.204374408	-0.370567887	0	0
-0.155552356	-0.173926992	-0.358016627	0	0

duration	srcbytes	dstbytes	land	wrongfragment
-0.156441757	0.914222239	-0.390900834	0	-0.142857143
-0.156441757	-0.042773551	-0.390900834	0	-0.142857143
-0.156441757	-0.447763074	-0.390900834	0	-0.142857143
-0.156441757	0.195781921	2.563320983	0	-0.142857143
-0.156441757	0.104243194	-0.238714747	0	-0.142857143
-0.156441757	-0.447763074	-0.390900834	0	-0.142857143
-0.156441757	-0.447763074	-0.390900834	0	-0.142857143
-0.156441757	-0.447763074	-0.390900834	0	-0.142857143
-0.156441757	-0.447763074	-0.390900834	0	-0.142857143

3.3. *Local Outliner Factor* (LOF) dan *Rule-based System Development*

Pada tahap ini diperlukan pembuatan model. Model yang digunakan adalah *Local Outliner Factor* (LOF). Pada tahap ini dilakukan pembuatan *environtment* sebagai tempat aliran data langsung. Nantinya, algoritma LOF akan bekerja sebagai pendeteksi anomali serangan jaringan melalui data yang digunakan. Pada tahap ini juga dilakukan pengklasifikasian jenis-jenis anomali jaringan yang digunakan sebagai parameter anomali jaringan sehingga model yang dirancang mampu mendeteksi parameter yang digunakan.

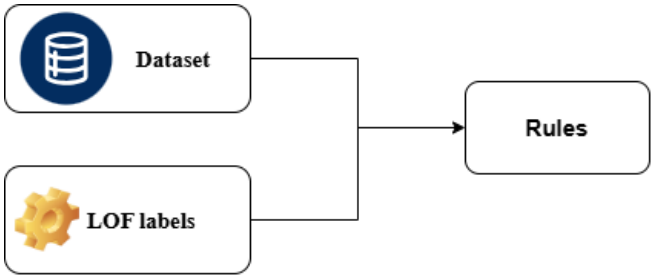


Sumber: [5]

Gambar 3.3 Contoh LOF

Gambar 3.3 menunjukkan bahwa LOF bekerja dengan mengevaluasi kepadatan lokal dari sebuah titik data dan membandingkannya dengan kepadatan dari titik-titik data di sekitarnya. Algoritma ini menghitung jarak ke tetangga-tetangga terdekat (*k-nearest neighbors*) untuk setiap titik, kemudian mengukur kepadatan lokal titik tersebut terhadap tetangganya. Jika kepadatan lokal titik jauh lebih rendah dibandingkan dengan tetangganya, maka titik tersebut dianggap sebagai *outlier*. Nilai LOF di atas 1 menunjukkan bahwa titik tersebut merupakan *outlier*, dan semakin tinggi nilainya, semakin besar kemungkinan titik tersebut adalah anomali. LOF sangat cocok digunakan dalam lingkungan dinamis seperti jaringan komputer, di mana pola data sering kali bervariasi dan sulit diprediksi.

Selanjutnya, penentuan *Rule-based System* untuk peningkatan keakuratan sistem deteksi yang telah dilakukan oleh model LOF. Pada tahap ini model memilah kembali anomali yang dideteksi oleh LOF sesuai dengan *rule* yang diimplementasikan pada model. *Rule* yang diimplementasikan merupakan hasil dari ekstraksi data yang diperoleh.



Gambar 3.4 Contoh *Rule-based System*

Gambar 3.4 menggambarkan alur sistem yang memanfaatkan kombinasi algoritma LOF dan *Rule-based System* dalam mendeteksi anomali pada sebuah *dataset*. Pada tahap awal, *dataset* yang berisi data dari lalu lintas jaringan atau data relevan lainnya diolah menggunakan LOF. Algoritma LOF bertugas memberikan label pada setiap data berdasarkan perhitungan kepadatan lokal dari titik data tersebut. Dengan membandingkan kepadatan lokal setiap titik dengan tetangga-tetangganya, LOF menghasilkan label yang mengindikasikan apakah data tersebut merupakan *outlier* atau data normal.

Setelah label-label LOF diberikan, data kemudian diproses lebih lanjut melalui sistem berbasis aturan. Sistem ini berfungsi untuk menerapkan aturan-aturan spesifik yang dirancang berdasarkan kondisi tertentu. Misalnya, jika label LOF menunjukkan bahwa suatu data merupakan anomali dan kondisi tertentu dalam *dataset* terpenuhi, maka data tersebut akan dikategorikan sebagai ancaman. Sistem berbasis aturan ini bertindak sebagai penguat, melengkapi hasil deteksi dari LOF dengan memberikan keputusan yang lebih presisi dan terarah.

3.4. *Output Development*

Pada tahap ini dilakukan pengembangan terhadap *output* sistem yang dibangun. Sistem ini dirancang dengan *output display* yang menggambarkan anomali yang terdeteksi serta menampilkan data dan informasi mengenai anomali maupun lalu lintas jaringan normal yang terdeteksi.

3.5. *Analysis of Result*

Hasil dari integrasi LOF dan *Rule-based System* akan dianalisis untuk menilai tingkat keakuratannya dalam mendeteksi serangan siber. Dengan demikian, akan diketahui tindakan tepat yang dapat dilakukan. Hasil dari analisis menjadi landasan untuk menyimpulkan efektivitas dari sistem yang dibangun.

BAB 4

HASIL DAN PEMBAHASAN

4.1. Implementasi Sistem dengan Bahasa Python

4.1.1. *Import library*

Library atau pustaka dalam Python adalah sekumpulan modul, fungsi, dan kelas yang telah dibuat oleh pengembang untuk memudahkan pemrograman. Penggunaan pustaka ini dilakukan untuk meningkatkan efisiensi dan memperluas kemampuan Python seperti membuat grafik, pemrosesan gambar, dan sebagainya. Gambar 4.1 merupakan skrip yang digunakan untuk memanggil pustaka yang akan digunakan.

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import LocalOutlierFactor
import matplotlib.pyplot as plt
```

Gambar 4.1 Pustaka Python

4.1.2. *Input data*

Data yang digunakan adalah dua buah *dataset* yang diperoleh dari Kaggle. Sebelum data di-*input* ke dalam sistem perlu dilakukannya *data schema alignment*, yaitu menyamakan dua *dataset* yang memiliki nama *tabel* berbeda namun nilai yang sama, hal ini diperlukan karena kedua *dataset* tersebut memiliki nama kolom yang berbeda namun memiliki nilai yang sama sehingga perlu dilakukan penyamaan skema data agar dapat dibaca oleh model sistem yang dibangun. Proses ini dilakukan dengan *mapping column* dan juga *renaming column.* Setelah dilakukannya penyamaan skema data selanjutnya adalah menambahkan skrip untuk *input dataset* ke dalam sistem model dengan, Gambar 4.2 adalah skrip yang digunakan untuk *input dataset*.

```
file_path1 = r"D:\Skripsian kita
ces\dataset_normalisasi\refined_network_anomaly_data.csv"
file_path2 = r"D:\Skripsian kita
ces\dataset_normalisasi\refined_network_anomaly_data_updated.csv"
```

Gambar 4.2 *Input Dataset*

Setelah data di-*input* ke dalam sistem model yang dibangun selanjutnya adalah menambahkan skrip untuk membaca *dataset* tersebut, sehingga sistem model yang dibangun dapat memproses data yang di-*input*. Skrip pada Gambar 4.3 digunakan untuk membaca kedua *dataset* tersebut.

```
data1 = pd.read_csv(file_path1, low_memory=False)
data2 = pd.read_csv(file_path2, low_memory=False)
```

Gambar 4.3 Pembaca *Dataset*

Berikut merupakan cuplikan data dari kedua *dataset* yang di-*input* pada sistem.

Tabel 4.1 Cuplikan *Dataset* 1 dan *Dataset* 2 setelah dilakukan *Schema Data Alignment*

duration	protocoltype	service	flag	srcbytes
0	tcp	ftp_data	SF	491
0	udp	other	SF	146
0	tcp	private	S0	0
0	tcp	http	SF	232
0	tcp	http	SF	199
0	tcp	private	REJ	0
0	tcp	private	S0	0
0	tcp	private	S0	0
0	tcp	remote_job	S0	0

duration	protocoltype	service	flag	srcbytes
0	udp	private	SF	105
0	udp	private	SF	105
0	udp	private	SF	105
0	udp	private	SF	105
0	udp	private	SF	105
0	udp	domain_u	SF	29
0	udp	private	SF	105
0	udp	private	SF	105
0	tcp	http	SF	223

4.1.3. *Pre-procecing data*

Pengimplementasian model perlu dilakukan pra-pemroresan data untuk memperoleh hasil maksimal, tahap pertama dalam pra-pemrosesan data ini dilakukan dengan mengombinasikan kedua *dataset* tersebut dengan menggunakan metode *Concatenation*

yakni menyusun *dataset* satu di bawah atau di samping *dataset* lainnya, metode ini menggunakan pustaka *PandaS* yang telah di *import*. Skrip pada Gambar 4.4 digunakan untuk mengombinasikan dua *dataset* tersebut.

```
combined_data = pd.concat([data1, data2], ignore_index=True)
```

Gambar 4.4 Kombinasi Data

Dataset yang sudah dikombinasikan perlu dipilih kolom yang mengandung nilai numerik untuk selanjutnya dilakukan standarisasi dengan metode *Scaling*. Gambar 4.5 adalah skrip yang digunakan untuk memilah kolom numerik.

```
numerical_features = combined_data.select_dtypes(include=[np.number])
```

Gambar 4.5 Pemilah Kolom Numerik

Standarisasi dilakukan dengan metode *Scale. Scale* pada *dataset* yang digunakan memiliki fungsi penting untuk memastikan semua fitur data numerik memiliki skala yang sama dan seimbang, sehingga algoritma *machine learning* dapat bekerja lebih optimal dan efisien. Gambar 4.6 merupakan skrip yang digunakan untuk *Scaling* pada *dataset* tersebut.

```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(numerical_features)
```

Gambar 4.6 Standarisasi *Scale*

4.1.4. *Implementasi Local Outlier Factor (LOF)*

Local Outlier Factor (LOF) adalah algoritma yang termasuk dalam metode pembelajaran tanpa pengawasan (*unsupervised learning*) yang digunakan untuk mendeteksi anomali berdasarkan kepadatan lokal data. LOF bekerja dengan menghitung skor *outlier* melalui perbandingan kepadatan lokal suatu titik data dengan kepadatan lokal tetangga-tetangganya. Titik data yang memiliki kepadatan jauh lebih rendah dibandingkan tetangganya akan dianggap sebagai *outlier* atau anomali.

```
lof = LocalOutlierFactor(n_neighbors=20, contamination=0.05)
combined_data['anomaly'] = lof.fit_predict(scaled_data)
```

Gambar 4.7 Implementasi LOF

Parameter `n_neighbors=20` menentukan jumlah tetangga terdekat yang digunakan untuk menghitung kepadatan lokal sebuah titik data. Semakin banyak tetangga yang dipertimbangkan, semakin akurat estimasi kepadatan lokal, meskipun ini

juga dapat meningkatkan waktu komputasi. Sementara itu, parameter `contamination=0.05` menunjukkan persentase sampel yang diharapkan menjadi anomali; dalam kasus ini, 5% dari sampel diasumsikan sebagai anomali, dan nilai ini membantu model menentukan ambang batas untuk mengklasifikasikan anomali.

Proses `fit_predict(scaled_data)` mengoperasikan dua fungsi. Pertama, `fit()` untuk mencocokkan model LOF pada data yang sudah di standardisasi dan menghitung kepadatan lokal berdasarkan parameter yang telah ditetapkan. Kedua, `predict()` untuk mengklasifikasikan titik-titik data sebagai *inlier* (1) atau *outlier* (-1). Hasil klasifikasi tersebut kemudian disimpan dalam kolom *anomaly* di dalam *DataFrame combined_data*, yang digunakan untuk mengidentifikasi dan menganalisis titik-titik data yang dianggap sebagai anomali.

Kemudian ditambahkan skrip filterisasi untuk mendeteksi dan memfilter data yang diklasifikasikan sebagai anomali seperti pada Gambar 4.8.

```
anomalies = combined_data[combined_data['anomaly'] == -1]
```

Gambar 4.8 Filterisasi

Dalam hal ini, baris-baris data yang diberi label -1 (yang menandakan bahwa mereka adalah *outlier* atau anomali) dipilih dan disimpan dalam sebuah *DataFrame* baru bernama *anomalies*. Data ini kemudian siap untuk dianalisis lebih lanjut guna memahami pola atau karakteristik yang mungkin menunjukkan penyimpangan dari data normal.

Setelah dilakukan filterisasi, ditambahkan klasifikasi terhadap jenis anomali yang dideteksi dengan menambahkan skrip pada Gambar 4.9.

```
def classify_risk(protocol):
    risk_levels = {
        'tcp': ('TCP Attack', 'red'),
        'udp': ('UDP Attack', 'orange'),
        'dos': ('DoS Attack', 'yellow'),
        'probing': ('Probing Attack', 'red'),
        'normal': ('Normal Traffic', 'green'),
        'unknown': ('Unknown Attack', 'black')
    }
    return risk_levels.get(protocol, ('Unknown Attack', 'black'))
```

Gambar 4.9 Klasifikasi LOF

Fungsi `classify_risk(protocol)` adalah fungsi yang dirancang untuk mengklasifikasikan tingkat risiko serangan jaringan berdasarkan jenis protokol yang digunakan. Fungsi ini menerima satu parameter bernama *protocol*, yang merupakan

string yang merepresentasikan jenis protokol jaringan seperti 'tcp', 'udp', dan sebagainya.

Di dalam fungsi ini, terdapat sebuah *dictionary* bernama `risk_levels` yang menyimpan pasangan *key-value*, di mana *key* adalah jenis protokol dan *value*-nya adalah *tuple* yang terdiri dari dua elemen: deskripsi serangan yang diidentifikasi dan warna yang diasosiasikan dengan tingkat risiko serangan tersebut. Misalnya, 'tcp': ('TCP *Attack*', '*red*') menunjukkan bahwa serangan jenis TCP diklasifikasikan sebagai serangan TCP dengan warna merah, yang menandakan tingkat risiko tinggi.

Isi dari *dictionary* `risk_levels` mengklasifikasikan berbagai jenis protokol dengan deskripsi dan warna yang berbeda. Protokol 'tcp' diasosiasikan dengan serangan TCP dan warna merah, sedangkan 'udp' dikaitkan dengan serangan UDP dan warna oranye. Protokol 'dos' dikategorikan sebagai serangan DoS dengan warna kuning, dan '*probing*' sebagai serangan *Probing* dengan warna merah. Jika protokolnya 'normal', maka ini dikategorikan sebagai lalu lintas normal dengan warna hijau, dan jika protokolnya '*unknown*', maka dianggap sebagai serangan yang tidak diketahui dengan warna hitam.

Fungsi ini menggunakan metode `.get()` dari *dictionary* untuk mencari nilai berdasarkan *key* yang diberikan, yakni jenis protokol. Jika protokol yang dimasukkan sesuai dengan salah satu *key* dalam *dictionary*, fungsi akan mengembalikan pasangan deskripsi serangan dan warna yang relevan. Jika protokol yang dimasukkan tidak ditemukan dalam *dictionary*, fungsi akan mengembalikan nilai *default* berupa *tuple* ('*Unknown Attack*', '*black*'), yang menunjukkan bahwa serangan tidak dikenal dan diwakili dengan warna hitam.

Fungsi ini sangat berguna dalam konteks keamanan jaringan, karena memungkinkan pengenalan cepat terhadap jenis serangan dan tingkat risikonya, serta memberikan visualisasi risiko melalui penggunaan warna yang diasosiasikan dengan jenis serangan tertentu.

4.1.5. *Implementasi Rule-based System*

Sistem berbasis aturan (R*ule-based System*) adalah sebuah pendekatan kecerdasan buatan yang menggunakan sekumpulan aturan yang telah ditentukan sebelumnya untuk membuat keputusan atau klasifikasi berdasarkan data yang dihadapi. Dalam konteks ini, sistem berbasis aturan digunakan untuk mengidentifikasi dan mengklasifikasikan risiko

pada data jaringan berdasarkan protokol dan karakteristik lalu lintas. Sistem ini bekerja dengan menetapkan aturan-aturan spesifik yang mendefinisikan kondisi tertentu dan respons yang harus diberikan ketika kondisi tersebut terpenuhi. Misalnya, jika jumlah *byte* yang dikirim melebihi ambang batas tertentu dan jenis protokol yang digunakan adalah TCP, maka sistem akan mengklasifikasikan data tersebut sebagai risiko tinggi. Pendekatan ini memungkinkan deteksi cepat terhadap potensi ancaman karena aturan yang digunakan bersifat langsung dan eksplisit, meskipun mungkin terbatas dalam menangani pola yang kompleks dan dinamis yang tidak tertangkap oleh aturan yang ada. Gambar 4.10 adalah skrip yang digunakan untuk implementasi *Rule-based System*.

```
def rule_based_classification(row):
    if row['protocoltype'] == 'tcp' and row['srcbytes'] > 1000:
        return 'High Risk', 'red'
    elif row['protocoltype'] == 'udp' and row['dstbytes'] > 1500:
        return 'Medium Risk', 'orange'
    elif row['protocoltype'] == 'dos' and row['count'] > 50:
        return 'High Risk', 'red'
    elif row['protocoltype'] == 'probing' and (row['srvcount'] < 50 or row['count']
> 20):
        return 'High Risk', 'red'
    elif row['protocoltype'] == 'normal':
        return 'Normal Traffic', 'green'
    else:
        return classify_risk('unknown')
```

Gambar 4.10 Klasifikasi *Rule-based System*

Pada fungsi `rule_based_classification`, setiap baris data yang diterima diklasifikasikan berdasarkan protokol yang digunakan dan metrik terkait lainnya, seperti jumlah *byte* sumber (*srcbytes*), jumlah *byte* tujuan (*dstbytes*), dan frekuensi kejadian (*count*). Misalnya, lalu lintas TCP dengan *srcbytes* lebih dari 1000 akan dikategorikan sebagai risiko tinggi, sedangkan protokol UDP dengan *dstbytes* melebihi 1500 dikategorikan sebagai risiko menengah. Demikian pula, serangan DoS dengan jumlah kejadian lebih dari 50 atau *Probing* dengan kondisi tertentu akan dikenali sebagai ancaman yang memerlukan perhatian khusus. Jika tidak ada kondisi yang sesuai dengan aturan yang ada, lalu lintas akan diidentifikasi sebagai lalu lintas normal atau serangan tidak dikenal. Sistem ini, meskipun sederhana, memberikan klasifikasi awal yang bermanfaat untuk pengawasan jaringan, memungkinkan identifikasi dini terhadap aktivitas abnormal yang dapat mengindikasikan potensi serangan.

Ambang batas dalam sistem deteksi anomali jaringan ditetapkan berdasarkan analisis pola lalu lintas normal dan anomali. Fitur seperti *srcbytes* dan *dstbytes* digunakan karena serangan *Denial of Service* (DoS) atau *flooding* biasanya melibatkan

pengiriman data besar dalam waktu singkat. Ambang batas 1000 *byte* dipilih karena ukuran data dalam serangan ini jauh lebih besar dibandingkan lalu lintas normal.

Protokol yang digunakan, seperti TCP dan UDP, juga mempengaruhi ambang batas. Misalnya, serangan UDP *flood* memanfaatkan protokol UDP, sedangkan TCP SYN *flood* menggunakan protokol TCP. Jumlah koneksi (dalam `count` dan `srvcount`) sangat relevan untuk mendeteksi serangan, dengan ambang batas seperti `count` > 50 atau `srvcount` < 50 ditetapkan berdasarkan pemahaman bahwa lalu lintas normal tidak menghasilkan koneksi sebanyak itu dalam waktu singkat.

Rasio kesalahan, seperti *srvserrorrate* dan *dsthostserrorrate*, juga digunakan untuk mengidentifikasi serangan yang menyebabkan banyak kegagalan koneksi, dengan rasio kesalahan tinggi menunjukkan adanya masalah serius. Nilai *flag* koneksi, seperti S0 dan SF, membantu mendeteksi serangan seperti SYN *flood*, di mana banyak koneksi tidak selesai.

Secara keseluruhan, ambang batas ini ditetapkan berdasarkan karakteristik spesifik jenis serangan dan analisis data historis untuk mendeteksi anomali dengan akurat dan efisien.

4.1.6. *Output*

Output sistem ini dirancang dengan menampilkan *display* dari anomali yang terdeteksi, lalu ditambahkan informasi mengenai anomali dan lalu lintas jaringan jika *dot* (titik) deteksi terkena kursor *mouse*.

```
fig, ax = plt.subplots(figsize=(8, 6))
```

Gambar 4.11 Inisiasi *Plot*

Pada awal skrip, dilakukan inisialisasi elemen visualisasi dengan menggunakan fungsi plt.subplots untuk menciptakan objek *figure* (`fig`) dan sumbu plot (`ax`). Parameter `figsize=(8, 6)` digunakan untuk menentukan ukuran grafik, dengan lebar 8 *inchi* dan tinggi 6 *inchi*. Hal ini bertujuan untuk memastikan visualisasi yang proporsional dan mudah dianalisis.

```
scatters = []
for index, row in combined_data.iterrows():
    risk, color = rule_based_classification(row)
    scatter = ax.scatter(index, row['srcbytes'], color=color)
    scatters.append((scatter, row))
```

Gambar 4.12 Pembuatan *Scatter Plot*

Proses iterasi dilakukan terhadap kumpulan data yang disimpan dalam `combined_data` menggunakan fungsi `iterrows()`. Setiap baris data kemudian diklasifikasikan berdasarkan aturan tertentu yang diterapkan melalui fungsi `rule_based_classification`. Fungsi ini menghasilkan dua keluaran utama, yaitu tingkat risiko (*risk level*) dan warna yang merepresentasikan hasil klasifikasi tersebut. Titik-titik data (*scatter*) kemudian digambarkan pada grafik menggunakan fungsi `ax.scatter`, dengan:

1. Sumbu-x merepresentasikan indeks pengamatan.
2. Sumbu-y menggambarkan jumlah *source bytes* (volume lalu lintas jaringan) yang dikirim oleh sumber.
3. Parameter warna menggambarkan klasifikasi risiko.

Setiap titik *scatter* disimpan ke dalam *list scatters* bersama dengan informasi baris data terkait. Penyimpanan tersebut bertujuan untuk memfasilitasi pengambilan informasi saat anotasi ditampilkan.

```
ax.set_title('LOF-based Network Anomaly Detection with Rule-Based System')
ax.set_xlabel('Index (Observation Points)')
ax.set_ylabel('Source Bytes (Traffic Volume)')
```

Gambar 4.13 Penambahan Judul

Judul grafik dan label sumbu ditambahkan untuk memberikan konteks kepada pembaca:

1. Judul "*LOF-based Network Anomaly Detection with Rule-Based System*" menjelaskan fokus dari visualisasi ini.
2. Label pada sumbu-x (*Index (Observation Points)*) dan sumbu-y (*Source Bytes (Traffic Volume)*) memberikan deskripsi singkat mengenai dimensi data.

```
annot = ax.annotate("", xy=(0, 0), xytext=(15, 15), textcoords="offset points",
bbox=dict(boxstyle="round", fc="w"), arrowprops=dict(arrowstyle="->"))
annot.set_visible(False)
```

Gambar 4.14 Inisiasi Anotasi

Sebuah objek anotasi dibuat menggunakan fungsi `ax.annotate` untuk memberikan informasi tambahan secara interaktif ketika pengguna mengarahkan kursor ke titik data tertentu. Parameter yang digunakan meliputi:

1. Posisi awal anotasi (`xy=(0, 0)`) dan jarak *offset* teks dari titik data (`xytext=(15, 15)`).
2. Gaya kotak teks menggunakan `bbox` dengan bentuk bundar dan warna latar putih.

3. Arah panah yang menunjuk ke titik data menggunakan `arrowstyle`.

Anotasi tersebut diinisialisasi dalam keadaan tidak terlihat menggunakan (`annot.set_visible(False)`), sehingga hanya akan tampil ketika interaksi terjadi.

```
def update_annot(ind):
    pos = scatters[ind][0].get_offsets()[0]
    annot.xy = pos
    row = scatters[ind][1]
    risk, _ = rule_based_classification(row)
      text = f"Protocol: {row['protocoltype']}\nRisk Level: {risk}\nSrcBytes:
{row['srcbytes']}"
    annot.set_text(text)
    annot.get_bbox_patch().set_facecolor(scatters[ind][0].get_facecolor()[0])
    annot.get_bbox_patch().set_alpha(0.6)
```

Gambar 4.15 Fungsi Memperbaharui Anotasi

Fungsi `update_annot` pada Gambar 4.15 dirancang untuk memperbarui isi dan posisi anotasi berdasarkan titik data yang sedang di-*hover* oleh pengguna. Proses pembaruan mencakup:

1. Penyesuaian posisi anotasi sesuai koordinat titik data.
2. Pengambilan informasi relevan dari baris data, seperti protokol jaringan (*protocoltype*), tingkat risiko (*risk level*), dan volume data (*srcbytes*).
3. Pengubahan teks anotasi sesuai informasi tersebut.
4. Penyesuaian warna latar belakang kotak teks agar konsisten dengan warna titik *scatter*.

Fungsi tersebut memastikan bahwa anotasi yang ditampilkan relevan dan sesuai dengan konteks data yang sedang diakses.

```
def hover(event):
    vis = annot.get_visible()
    if event.inaxes == ax:
        for i, (scatter, row) in enumerate(scatters):
            cont, _ = scatter.contains(event)
            if cont:
                update_annot(i)
                annot.set_visible(True)
                fig.canvas.draw_idle()
                return
    if vis:
        annot.set_visible(False)
        fig.canvas.draw_idle()
```

Gambar 4.16 Interaksi *Hover*

Fungsi *hover* pada Gambar 4.16 digunakan untuk menangkap gerakan kursor pada area *plot*. Ketika kursor berada di atas sebuah titik data, fungsi tersebut:

1. Memeriksa apakah kursor berada di dalam area plot (`event.inaxes == ax`).

2. Memastikan interaksi antara kursor dan titik data melalui metode `scatter.contains(event)`.
3. Jika interaksi terjadi, fungsi `update_annot` dipanggil untuk memperbarui anotasi, dan anotasi ditampilkan dengan `annot.set_visible(True)`.
4. Jika tidak ada interaksi, anotasi disembunyikan kembali.

Metode `fig.canvas.draw_idle()` pada Gambar 4.16 digunakan untuk memastikan bahwa perubahan pada anotasi segera direfleksikan dalam grafik.

```
fig.canvas.mpl_connect("motion_notify_event", hover)
```

Gambar 4.17 Menghubungkan *Event Hover*

Pada skrip di Gambar 4.17 digunakan fungsi `mpl_connect` untuk menghubungkan *event listener* terhadap grafik. *Event* yang dipantau adalah `motion_notify_event`, yaitu perubahan posisi kursor. Fungsi *hover* secara otomatis dipanggil ketika *event* tersebut terdeteksi, memungkinkan sistem merespons secara *real-time* terhadap interaksi pengguna.

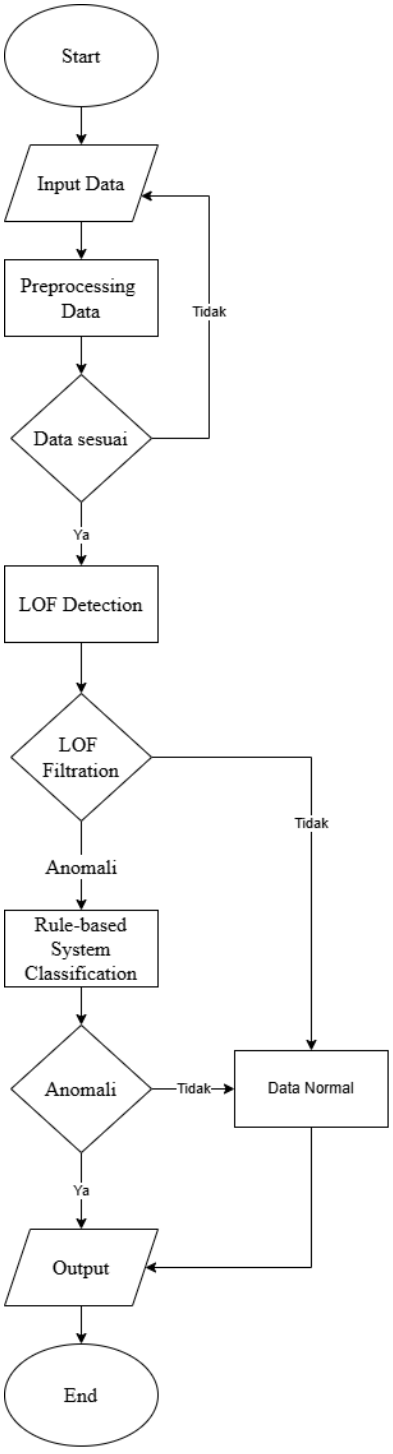
```
plt.show()
```

Gambar 4.18 Menampilkan Antarmuka

Akhirnya, visualisasi ditampilkan menggunakan fungsi `plt.show()`, yang membuka antarmuka interaktif sehingga pengguna dapat menjelajahi grafik dan melihat detail setiap titik data melalui anotasi.

4.2. Pengujian Program

Kombinasi dari *Local Outlier Factor* (LOF) dan *Rule-based System* melewati beberapa proses seperti pada Gambar 4.19, proses yang dijalankan oleh sistem yang dibangun memiliki kompleksitas yang cukup sehingga hasil deteksi yang dihasilkan cukup akurat dalam membentuk mitigasi serangan jaringan yang dapat merusak infrastruktur jaringan.



Gambar 4.19 Alur Proses Sistem

Adapun penjelasan mengenai proses sistem adalah sebagai berikut.

1. Mulai (*Start*)

   Tahap awal ini merupakan proses inisialisasi sistem, di mana semua komponen yang diperlukan untuk mendeteksi anomali jaringan diaktifkan. Inisialisasi ini mencakup pengaturan parameter yang akan digunakan dalam algoritma LOF serta penyiapan

aturan klasifikasi berbasis *Rule-based System*. Inisialisasi bertujuan agar sistem siap untuk menerima data *input* dan melakukan deteksi serta analisis terhadap pola lalu lintas jaringan.

2. *Input* Data

   Sistem menerima data lalu lintas jaringan sebagai *input* utama yang akan dianalisis. Data ini bisa berupa *dataset* historis yang sudah ada atau data jaringan *real* (nyata) yang ditangkap secara langsung dengan menggunakan aplikasi pihak ketiga seperti Wireshark, yang mampu menangkap paket jaringan dalam format yang dapat diproses oleh algoritma deteksi. Pada tahap ini, data yang diterima biasanya terdiri dari berbagai atribut jaringan, seperti alamat IP sumber dan tujuan, jumlah *byte* yang dikirim atau diterima, jenis protokol (misalnya, TCP atau UDP), waktu kedatangan paket, dan atribut lain yang penting untuk mendeteksi pola anomali. Keakuratan dan kelengkapan data pada tahap ini sangat penting karena akan mempengaruhi hasil deteksi anomali.

3. *Pre-processing* Data

   Tahap *pre-processing* adalah proses penyiapan data agar bisa dianalisis dengan optimal oleh algoritma LOF. *Pre-processing* biasanya melibatkan beberapa langkah penting, antara lain:

   a. Penyamaan skema data. Data harus melalui penyemaan skema data agar berada dalam rentang yang sama, karena algoritma LOF sensitif terhadap perbedaan skala pada atribut data. Misalnya, atribut yang memiliki rentang nilai besar (seperti jumlah *byte*) dapat mendominasi atribut lainnya, sehingga penyamaan skema data diperlukan agar setiap fitur memiliki bobot yang seimbang.
   b. Standarisasi. Data diubah menjadi bentuk numerik yang sesuai untuk proses analisis. Data kategori atau teks diubah menjadi nilai numerik, dan atribut yang tidak relevan atau *redundant* dapat dibuang untuk meningkatkan efisiensi.

4. LOF *Detection*

   Pada tahap ini, algoritma LOF digunakan untuk mendeteksi anomali dalam data yang sudah di pre-proses. LOF adalah algoritma berbasis kepadatan yang menghitung kepadatan lokal dari titik data dan membandingkannya dengan kepadatan data di sekitarnya. Langkah-langkah dalam deteksi LOF meliputi:

   a. Menghitung *Local Reachability Density*. LOF menghitung kepadatan lokal tiap titik data dengan memperhatikan jarak ke tetangga-tetangga terdekatnya.

b. Mengukur Skor LOF. Skor LOF dihitung untuk menentukan seberapa jauh sebuah titik berbeda dari lingkungan sekitarnya. Skor LOF yang lebih tinggi menunjukkan bahwa titik tersebut kemungkinan besar merupakan anomali.
c. Pemberian Label. Data yang memiliki skor LOF tinggi dianggap sebagai anomali dan diberi label khusus (-1) untuk membedakannya dari data normal. Dengan metode ini, LOF dapat mendeteksi anomali dengan lebih baik dibandingkan metode deteksi berbasis jarak yang tidak mempertimbangkan kepadatan lokal.

5. LOF *Filtration*

   Hasil deteksi dari algoritma LOF kemudian di-*filter* untuk menyaring data yang telah teridentifikasi sebagai anomali. Tahapan ini berfungsi untuk memastikan bahwa hanya data yang memiliki label anomali (-1) yang akan diteruskan ke tahap berikutnya. Data dengan label normal tidak akan diproses lebih lanjut, sehingga mengurangi jumlah data yang perlu diproses di tahap klasifikasi. Filtrasi ini juga membantu memfokuskan analisis pada data yang dianggap berpotensi sebagai ancaman, meningkatkan efisiensi sistem secara keseluruhan.

6. Klasifikasi *Rule-based System*

   Setelah data anomali berhasil di-*filter*, sistem mengklasifikasikan data tersebut berdasarkan aturan atau *rules* yang telah ditentukan. Proses ini mencakup identifikasi pola spesifik dan penentuan tingkat risiko berdasarkan kriteria yang telah ditetapkan. Berikut adalah beberapa aturan yang diterapkan dalam sistem:

   a. Klasifikasi berdasarkan Protokol TCP. Jika data menunjukkan aktivitas dengan protokol TCP dan nilai atribut *srcbytes* (jumlah *byte* sumber) lebih dari 1000, maka data tersebut dikategorikan sebagai risiko tinggi dan dianggap sebagai potensi serangan TCP *Attack*.
   b. Klasifikasi berdasarkan Protokol UDP. Jika atribut *dstbytes* (jumlah *byte* tujuan) pada data lebih dari 1500 dalam protokol UDP, data tersebut dikategorikan sebagai risiko menengah, yang menunjukkan potensi UDP *Attack*.
   c. Deteksi DoS (*Denial of Service*) dan *Probing*: Sistem juga dapat mendeteksi serangan DoS dan *Probing* berdasarkan pola tertentu, misalnya jika jumlah kejadian yang sama dalam rentang waktu tertentu melebihi ambang batas. Setiap data yang memenuhi aturan ini akan diberi label tingkat risiko, seperti rendah, menengah, atau tinggi, yang nantinya ditampilkan dalam *output*.

7. *Output*

   Setelah proses klasifikasi selesai, sistem menghasilkan *output* berupa visualisasi grafik. Visualisasi ini membantu pengguna untuk memahami pola anomali secara keseluruhan dan tingkat risiko dari setiap serangan yang terdeteksi. Jika ditemukan anomali dengan risiko tinggi, sistem akan menampilkan informasi lalu lintas jaringan kepada pengguna ketika kursor *mouse* diarahkan kepada titik yang di visualisasikan sistem. Informasi tersebut mencakup jenis serangan, tingkat risiko, dan data pendukung yang relevan. Fitur ini bertujuan untuk memberikan peringatan secara *real-time* agar pengguna dapat segera mengambil tindakan pencegahan.

8. Selesai (*End*)

   Tahap akhir pada Gambar 4.19 adalah mengembalikan sistem ke keadaan siap awal, di mana sistem akan menunggu data baru atau *input* berikutnya untuk diproses. Hal ini memungkinkan proses deteksi berjalan secara kontinu atau *looping*, sehingga sistem dapat terus memonitor lalu lintas jaringan dan mendeteksi anomali secara *real-time*. Sistem siap untuk melakukan deteksi baru setelah siklus ini selesai.

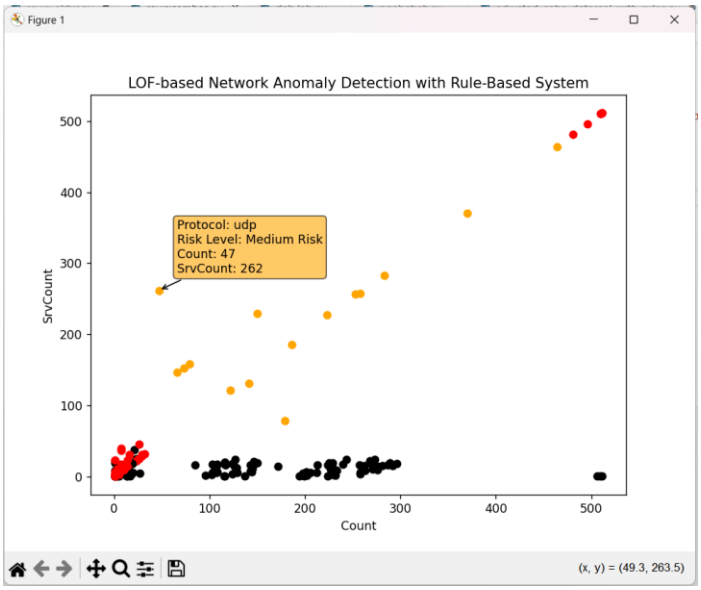
4.2.1. *Pengujian program dengan dataset*

Untuk dapat mengevaluasi program yang telah dibuat dilakukanlah proses pengujian program, dengan tujuan memastikan pendeteksian anomali menggunakan algoritma *Local Outlier Factor* (LOF) dan *Rule-based System* yang dilakukan berjalan dengan baik dan akurat. Proses pengujian ini menggunakan *dataset* yang telah dikumpulkan kemudian dilakukannya penyamaan skema data. *Dataset* tersebut dipilih karena mengandung informasi data anomali serangan jaringan seperti TCP *Flood*, UDP *Flood*, *Probing*, dan DoS serta informasi lain yang dibutuhkan dalam proses deteksi seperti *srcbytes*, *protocoltype* dan sebagainya.

Sebelum program di-*run*, perlu diperhatikan *path dataset* yang di-*input* pada program sesuai dengan lokasi *dataset* yang disimpan pada direktori lokal komputer yang digunakan, setelah program di-*run* maka muncul tampilan seperti Gambar 4.20.

```
Processing: tcp - srcbytes: 2599, dstbytes: 293, count: 3, srvcount: 3
Processing: tcp - srcbytes: 2599, dstbytes: 293, count: 3, srvcount: 3
Processing: tcp - srcbytes: 2599, dstbytes: 293, count: 2, srvcount: 2
Processing: tcp - srcbytes: 2599, dstbytes: 293, count: 2, srvcount: 2
Processing: tcp - srcbytes: 2599, dstbytes: 293, count: 3, srvcount: 3
Processing: tcp - srcbytes: 2599, dstbytes: 293, count: 3, srvcount: 3
Processing: tcp - srcbytes: 2599, dstbytes: 293, count: 3, srvcount: 3
Processing: tcp - srcbytes: 2599, dstbytes: 293, count: 2, srvcount: 2
Processing: tcp - srcbytes: 2599, dstbytes: 293, count: 2, srvcount: 2
Processing: tcp - srcbytes: 2599, dstbytes: 293, count: 3, srvcount: 3
Processing: tcp - srcbytes: 2599, dstbytes: 293, count: 2, srvcount: 2
Processing: tcp - srcbytes: 2599, dstbytes: 293, count: 2, srvcount: 2
Processing: udp - srcbytes: 29, dstbytes: 29, count: 3, srvcount: 5
Processing: udp - srcbytes: 29, dstbytes: 29, count: 3, srvcount: 5
Processing: udp - srcbytes: 29, dstbytes: 29, count: 3, srvcount: 5
```

Gambar 4.20 Proses Pembacaan *Dataset*



Gambar 4.21 Visualisasi Anomali menggunakan *Dataset*

Pada Gambar 4.21, diagram *Local Outlier Factor* (LOF) berfungsi sebagai visualisasi dari hasil deteksi anomali yang teridentifikasi dalam *dataset* jaringan. Diagram ini menggunakan dua sumbu untuk menggambarkan hubungan antara indeks observasi dan volume lalu lintas data.

Sumbu horizontal (sumbu-x) merepresentasikan jumlah *count* setiap titik data dalam *dataset* yang dianalisis oleh algoritma LOF. Sementara itu, sumbu vertikal (sumbu-y) menggambarkan jumlah *srvcount* yang terkirim dalam setiap titik data. Nilai ini merupakan salah satu parameter penting dalam mendeteksi anomali, di mana lonjakan atau penurunan signifikan pada jumlah *byte* dapat mengindikasikan potensi serangan jaringan.

Setiap titik pada diagram mewakili satu data, dengan warna yang berbeda-beda untuk mengindikasikan tingkat risiko yang terdeteksi. Titik-titik dengan warna hijau

merepresentasikan lalu lintas jaringan yang normal, di mana nilai LOF menunjukkan bahwa titik tersebut tidak mencurigakan atau tidak menandakan adanya anomali. Namun pada *dataset* yang digunakan untuk pengujian sistem ini, tidak terdapat lalu lintas jaringan normal sehingga titik hijau sebagai indikator lalu lintas jaringan normal tersebut tidak muncul pada visualisasi LOF yang dihasilkan. Titik-titik dengan warna kuning menunjukkan risiko menengah, di mana kepadatan lokal titik tersebut berbeda dari titik-titik di sekitarnya, namun tidak cukup signifikan untuk dianggap sebagai ancaman besar. Sedangkan, titik-titik berwarna merah menandakan adanya anomali yang signifikan atau serangan jaringan, di mana algoritma LOF telah mengidentifikasi titik tersebut sebagai *outlier*, yaitu titik dengan kepadatan lokal yang jauh lebih rendah dibandingkan dengan tetangganya.

Informasi serangan jaringan akan muncul apabila kursor *mouse* diarahkan ke titik yang dianggap sebagai anomali atau *outlier*, informasi yang diberikan tersebut mungkin memerlukan tindakan lebih lanjut. Secara keseluruhan, visualisasi ini memberikan gambaran yang jelas mengenai distribusi dan karakteristik data normal serta anomali, serta membantu dalam mengidentifikasi pola serangan jaringan yang mungkin terjadi. Hal ini sangat penting untuk mendukung keputusan mitigasi dini terhadap potensi ancaman siber seperti serangan *Denial of Service* (DoS) atau *flooding* pada jaringan.

Dengan demikian, diagram ini tidak hanya membantu dalam menganalisis hasil deteksi anomali, tetapi juga menjadi alat yang krusial dalam memahami distribusi serangan dan mengembangkan langkah mitigasi yang lebih responsif terhadap berbagai jenis serangan jaringan.

Setelah tampilan tersebut muncul pada layar komputer dipastikan bahwa program deteksi anomali jaringan menggunakan *Local Outlier Factor* (LOF) dan *Rule-based System* telah berjalan dengan baik. Dengan hasil yang diperoleh dari kombinasi kedua algoritma tersebut, mitigasi ancaman serangan jaringan yang berpotensi merusak infrastruktur jaringan dapat dilakukan.

4.2.2. *Pengujian program dengan data real*

Pengujian menggunakan data *real* adalah pengujian yang menggunakan data yang diambil langsung menggunakan bantuan pihak ketiga. Pada pengujian ini, data diambil

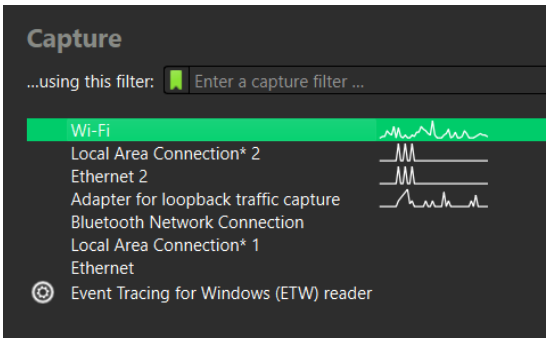
dari aplikasi Wireshark pada perangkat laptop yang terhubung dengan internet melalui konektifitas *Wi-Fi*.

Sumber: Wikimedia

Gambar 4.22 Wireshark

Wireshark merupakan alat analisis jaringan yang digunakan untuk memantau dan menganalisis lalu lintas jaringan internet. Dengan Wireshark, pengguna dapat menangkap (*capture*) berbagai jenis lalu lintas jaringan, seperti LAN, *Wi-Fi*, dan *Bluetooth*, serta memeriksa detail setiap paket yang ditransmisikan, termasuk alamat IP sumber dan tujuan, *port*, protokol yang digunakan, serta ukuran dan isi paket. Selain itu, Wireshark memungkinkan pengguna untuk mengidentifikasi masalah jaringan, memeriksa potensi ancaman keamanan, dan menganalisis performa jaringan secara mendalam.

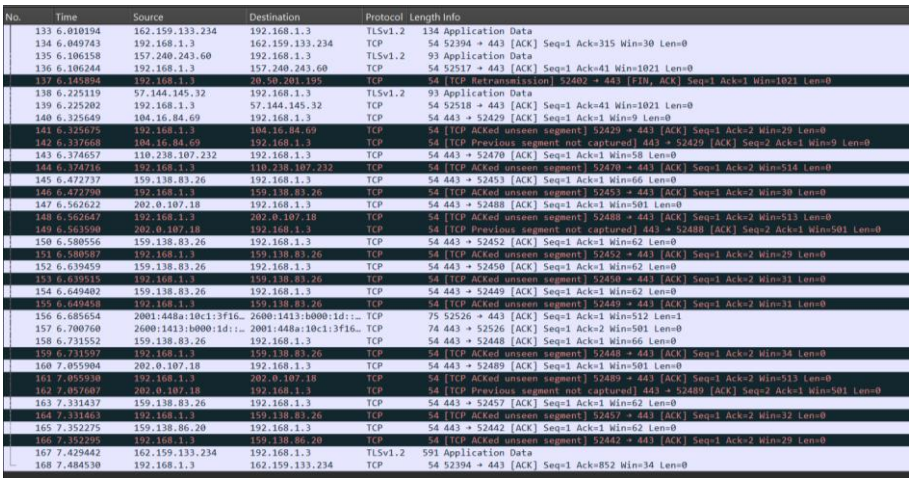
Pengambilan sampel data *real* menggunakan Wireshark dimulai dengan membuka aplikasi Wireshark pada perangkat yang digunakan. Setelah aplikasi terbuka, pengguna akan dihadapkan pada antarmuka utama yang menampilkan daftar jaringan yang tersedia. Dari daftar ini, pengguna dapat memilih jaringan yang ingin di-*capture* untuk analisis lebih lanjut. Pengambilan sampel data ini dilakukan di dalam lingkungan jaringan normal.



Gambar 4.23 Pemilihan Jaringan

Setelah memilih jaringan yang akan di-*capture*, pengguna akan melihat tampilan antarmuka Wireshark yang menampilkan berbagai informasi dan opsi analisis. Dalam

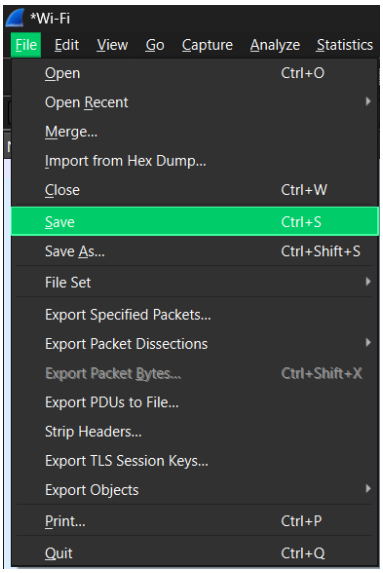
penelitian ini, data sampel yang berhasil dikumpulkan berjumlah 78.414 baris, mencerminkan aktivitas jaringan selama periode pengamatan yang ditentukan. Pengumpulan data *real* dilakukan pada tanggal 31 Oktober 2024, antara pukul 14:50 hingga 15:00 WIB.



No.	Time	Source	Destination	Protocol	Length	Info
133	6.010194	162.159.133.234	192.168.1.3	TLSv1.2	134	Application Data
134	6.049743	192.168.1.3	162.159.133.234	TCP	54	52394 → 443 [ACK] Seq=1 Ack=315 Win=30 Len=0
135	6.106158	157.240.243.60	192.168.1.3	TLSv1.2	93	Application Data
136	6.106244	192.168.1.3	157.240.243.60	TCP	54	52517 → 443 [ACK] Seq=1 Ack=41 Win=1021 Len=0
137	6.145894	192.168.1.3	20.50.201.195	TCP	54	[TCP Retransmission] 52402 → 443 [FIN, ACK] Seq=1 Ack=1 Win=1021 Len=0
138	6.225119	57.144.145.32	192.168.1.3	TLSv1.2	93	Application Data
139	6.225202	192.168.1.3	57.144.145.32	TCP	54	52518 → 443 [ACK] Seq=1 Ack=41 Win=1021 Len=0
140	6.325649	104.16.84.69	192.168.1.3	TCP	54	443 → 52429 [ACK] Seq=1 Ack=1 Win=9 Len=0
141	6.325675	192.168.1.3	104.16.84.69	TCP	54	[TCP ACKed unseen segment] 52429 → 443 [ACK] Seq=1 Ack=2 Win=29 Len=0
142	6.337668	104.16.84.69	192.168.1.3	TCP	54	[TCP Previous segment not captured] 443 → 52429 [ACK] Seq=2 Ack=1 Win=9 Len=0
143	6.374657	110.238.107.232	192.168.1.3	TCP	54	443 → 52470 [ACK] Seq=1 Ack=1 Win=58 Len=0
144	6.374716	192.168.1.3	110.238.107.232	TCP	54	[TCP ACKed unseen segment] 52470 → 443 [ACK] Seq=1 Ack=2 Win=514 Len=0
145	6.472737	159.138.83.26	192.168.1.3	TCP	54	443 → 52453 [ACK] Seq=1 Ack=1 Win=66 Len=0
146	6.472790	192.168.1.3	159.138.83.26	TCP	54	[TCP ACKed unseen segment] 52453 → 443 [ACK] Seq=1 Ack=2 Win=30 Len=0
147	6.562622	202.0.107.18	192.168.1.3	TCP	54	443 → 52488 [ACK] Seq=1 Ack=1 Win=501 Len=0
148	6.562647	192.168.1.3	202.0.107.18	TCP	54	[TCP ACKed unseen segment] 52488 → 443 [ACK] Seq=1 Ack=2 Win=513 Len=0
149	6.563590	202.0.107.18	192.168.1.3	TCP	54	[TCP Previous segment not captured] 443 → 52488 [ACK] Seq=2 Ack=1 Win=501 Len=0
150	6.580556	159.138.83.26	192.168.1.3	TCP	54	443 → 52452 [ACK] Seq=1 Ack=1 Win=62 Len=0
151	6.580587	192.168.1.3	159.138.83.26	TCP	54	[TCP ACKed unseen segment] 52452 → 443 [ACK] Seq=1 Ack=2 Win=29 Len=0
152	6.639459	159.138.83.26	192.168.1.3	TCP	54	443 → 52450 [ACK] Seq=1 Ack=1 Win=62 Len=0
153	6.639515	192.168.1.3	159.138.83.26	TCP	54	[TCP ACKed unseen segment] 52450 → 443 [ACK] Seq=1 Ack=2 Win=31 Len=0
154	6.649402	159.138.83.26	192.168.1.3	TCP	54	443 → 52449 [ACK] Seq=1 Ack=1 Win=62 Len=0
155	6.649458	192.168.1.3	159.138.83.26	TCP	54	[TCP ACKed unseen segment] 52449 → 443 [ACK] Seq=1 Ack=2 Win=31 Len=0
156	6.685654	2001:448a:10c1:3f16…	2600:1413:b000:1d::…	TCP	75	52526 → 443 [ACK] Seq=1 Ack=1 Win=512 Len=1
157	6.700760	2600:1413:b000:1d::…	2001:448a:10c1:3f16…	TCP	74	443 → 52526 [ACK] Seq=1 Ack=2 Win=501 Len=0
158	6.731552	159.138.83.26	192.168.1.3	TCP	54	443 → 52448 [ACK] Seq=1 Ack=1 Win=66 Len=0
159	6.731597	192.168.1.3	159.138.83.26	TCP	54	[TCP ACKed unseen segment] 52448 → 443 [ACK] Seq=1 Ack=2 Win=34 Len=0
160	7.055904	202.0.107.18	192.168.1.3	TCP	54	443 → 52489 [ACK] Seq=1 Ack=1 Win=501 Len=0
161	7.055930	192.168.1.3	202.0.107.18	TCP	54	[TCP ACKed unseen segment] 52489 → 443 [ACK] Seq=1 Ack=2 Win=513 Len=0
162	7.057607	202.0.107.18	192.168.1.3	TCP	54	[TCP Previous segment not captured] 443 → 52489 [ACK] Seq=2 Ack=1 Win=501 Len=0
163	7.331437	159.138.83.26	192.168.1.3	TCP	54	443 → 52457 [ACK] Seq=1 Ack=1 Win=62 Len=0
164	7.331463	192.168.1.3	159.138.83.26	TCP	54	[TCP ACKed unseen segment] 52457 → 443 [ACK] Seq=1 Ack=2 Win=32 Len=0
165	7.352275	159.138.86.20	192.168.1.3	TCP	54	443 → 52442 [ACK] Seq=1 Ack=1 Win=62 Len=0
166	7.352295	192.168.1.3	159.138.86.20	TCP	54	[TCP ACKed unseen segment] 52442 → 443 [ACK] Seq=1 Ack=2 Win=29 Len=0
167	7.429442	162.159.133.234	192.168.1.3	TLSv1.2	591	Application Data
168	7.484530	192.168.1.3	162.159.133.234	TCP	54	52394 → 443 [ACK] Seq=1 Ack=852 Win=34 Len=0

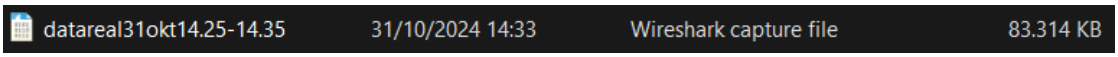
Gambar 4.24 Tampilan Lalu Lintas Jaringan di Wireshark

Setelah proses pengumpulan sampel data selesai, langkah selanjutnya adalah menyimpan data yang telah dikumpulkan. Untuk melakukannya, pengguna dapat menekan tombol ***File*** yang terletak di sudut kanan atas antarmuka Wireshark. Dari menu yang muncul, pengguna memilih opsi *Save* lalu memberi nama pada *File*. Dengan cara ini, data akan disimpan dalam format .pcapng, yang memudahkan analisis dan pengolahan data lebih lanjut.



Gambar 4.25 Menyimpan Data dari Wireshark

Setelah data berhasil disimpan maka akan menampilkan data seperti Gambar 4.25.



Gambar 4.26 Data *Real*

Selanjutnya, data tersebut di proses dengan menggunakan pyshark di Python dengan menggunakan skrip berikut:

```
import pandas as pd
import pyshark
import logging
from datetime import datetime
import json

# Configure logging with timestamp
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s'
)

def extract_layer_fields(packet, layer_name):
    """
    Extract all available fields from a specific packet layer.

    Args:
        packet: pyshark packet object
        layer_name: name of the layer to extract fields from

    Returns:
        dict: Dictionary containing all fields from the layer
    """
    fields = {}
    if hasattr(packet, layer_name):
        layer = getattr(packet, layer_name)
        for field_name in layer.field_names:
            try:
                fields[f'{layer_name}_{field_name}'] = getattr(layer,
field_name)
                if hasattr(layer, f'get_field_value'):
                    raw_value = layer.get_field_value(field_name, raw=True)
                    fields[f'{layer_name}_{field_name}_raw'] = raw_value
            except AttributeError:
                continue
    return fields

def get_packet_data(packet):
    packet_data = {
        'frame_number': packet.frame_info.number,
        'time': packet.frame_info.time,
        'time_epoch': packet.frame_info.time_epoch,
        'length': packet.length,
        'capture_length': packet.captured_length,
        #'interface_id': packet.interface_id,
        'highest_layer': packet.highest_layer,
```

```
        #'layers': ','.join(packet.layers_name)
    }

    # Extract fields from common layers
    common_layers = ['eth', 'ip', 'ipv6', 'tcp', 'udp', 'http', 'dns',
'icmp',
                     'arp', 'ssl', 'tls', 'smtp', 'ftp', 'dhcp']

    for layer in common_layers:
        layer_fields = extract_layer_fields(packet, layer)
        packet_data.update(layer_fields)

    return packet_data
def convert_pcap_to_detailed_csv(pcap_file_path, csv_output_path=None):

    if csv_output_path is None:
        timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
        csv_output_path = f"detailed_network_data_{timestamp}.csv"

    logging.info(f"Starting to process PCAP file: {pcap_file_path}")

    try:
        cap = pyshark.FileCapture(pcap_file_path)
        data_list = []
        packet_count = 0
        all_fields = set()

        logging.info("First pass: Collecting all possible fields...")
        for packet in cap:
            try:
                packet_data = get_packet_data(packet)
                all_fields.update(packet_data.keys())
                packet_count += 1

                if packet_count % 1000 == 0:
                    logging.info(f"Processed {packet_count} packets in first
pass...")

            except Exception as e:
                logging.warning(f"Error processing packet {packet_count +
1}: {e}")
                continue


        cap.reset()
        packet_count = 0
        data_list = []

        logging.info("Second pass: Extracting all field data...")
        for packet in cap:
            try:
                packet_data = get_packet_data(packet)

                for field in all_fields:
                    if field not in packet_data:
                        packet_data[field] = None

                data_list.append(packet_data)
```

```
                packet_count += 1

                if packet_count % 1000 == 0:
                    logging.info(f"Processed {packet_count} packets in
second pass...")

            except Exception as e:
                logging.warning(f"Error processing packet {packet_count +
1}: {e}")
                continue

        df = pd.DataFrame(data_list)



        df.to_csv(csv_output_path, index=False)

        logging.info(f"Successfully processed {packet_count} packets")
        logging.info(f"CSV file saved to: {csv_output_path}")

        return csv_output_path

    except Exception as e:
        logging.error(f"Failed to process PCAP file: {e}")
        raise


if __name__ == "__main__":
    pcap_file = "D:\datareal31okt14.25-14.35.pcapng"

    try:
        output_path = convert_pcap_to_detailed_csv(pcap_file)
        print("\nConversion completed successfully!")
        print(f"Output file: {output_path}")

    except Exception as e:
        print(f"\nConversion failed: {e}")
```

Program tersebut dirancang untuk memproses *file Packet Capture* (PCAP) dan mengonversinya menjadi *file Comma-Separated Values* (CSV) yang memuat informasi mendetail tentang setiap paket jaringan. Program tersebut memanfaatkan pustaka pyshark untuk membaca data dari *file* PCAP dan *PandaS* untuk mengolah serta menyimpan data dalam format CSV. Tujuan utama dari program tersebut adalah mempermudah analisis data jaringan dengan mengekstraksi seluruh informasi dari paket, termasuk data pada berbagai *layer* protokol seperti *Ethernet*, IP, TCP, UDP, HTTP, DNS, dan protokol lainnya.

```
2024-12-26 18:20:14,803 - INFO - Starting to process PCAP file: D:\datareal31okt14.25-14.35.pcapng
2024-12-26 18:20:14,804 - INFO - First pass: Collecting all possible fields...
2024-12-26 18:20:20,360 - INFO - Processed 1000 packets in first pass...
2024-12-26 18:20:24,734 - INFO - Processed 2000 packets in first pass...
2024-12-26 18:20:28,904 - INFO - Processed 3000 packets in first pass...
2024-12-26 18:20:33,027 - INFO - Processed 4000 packets in first pass...
2024-12-26 18:20:37,594 - INFO - Processed 5000 packets in first pass...
2024-12-26 18:20:41,838 - INFO - Processed 6000 packets in first pass...
2024-12-26 18:20:46,277 - INFO - Processed 7000 packets in first pass...
2024-12-26 18:20:50,654 - INFO - Processed 8000 packets in first pass...
2024-12-26 18:20:55,173 - INFO - Processed 9000 packets in first pass...
2024-12-26 18:20:59,577 - INFO - Processed 10000 packets in first pass...
2024-12-26 18:21:03,796 - INFO - Processed 11000 packets in first pass...
2024-12-26 18:21:08,177 - INFO - Processed 12000 packets in first pass...
2024-12-26 18:21:12,479 - INFO - Processed 13000 packets in first pass...
```

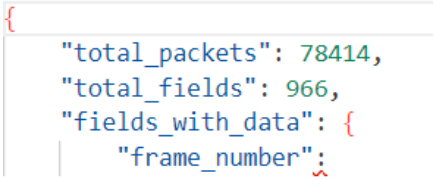
Gambar 4.27 Proses Tahap Pertama

Proses kerja program melibatkan dua tahapan utama. Pada tahap pertama, program membaca seluruh paket dalam *file* PCAP untuk mengidentifikasi dan mengumpulkan seluruh bidang (*fields*) yang mungkin ada. Informasi ini digunakan untuk memastikan bahwa setiap bidang yang ditemukan dapat diekstraksi pada tahap berikutnya. Pada tahap kedua, program mengekstraksi data secara mendetail dari setiap paket, termasuk informasi dasar seperti nomor *frame*, *timestamp*, panjang paket, dan *layer* tertinggi yang terdeteksi. Data dari setiap *layer* protokol juga diekstraksi dan disimpan dalam format struktur data Python berupa *dictionary*.

```
2024-12-26 18:26:13,423 - INFO - Second pass: Extracting all field data...
2024-12-26 18:26:19,094 - INFO - Processed 1000 packets in second pass...
2024-12-26 18:26:23,877 - INFO - Processed 2000 packets in second pass...
2024-12-26 18:26:28,406 - INFO - Processed 3000 packets in second pass...
2024-12-26 18:26:33,399 - INFO - Processed 4000 packets in second pass...
2024-12-26 18:26:38,327 - INFO - Processed 5000 packets in second pass...
```

Gambar 4.28 Proses Tahap Kedua

Selain menghasilkan *file* CSV, program juga menghitung statistik terkait jumlah total paket yang diproses, jumlah total bidang yang ditemukan, dan jumlah paket yang memiliki data pada setiap bidang tertentu. Statistik ini disimpan dalam *file* JSON sebagai pelengkap hasil analisis.



Gambar 4.29 *File* JSON Hasil Ekstraksi

Untuk memantau jalannya proses, program menggunakan *logging* untuk mencatat informasi seperti jumlah paket yang telah diproses serta peringatan jika terjadi kesalahan selama pemrosesan.

```
2024-12-26 18:33:50,280 - INFO - Successfully processed 78414 packets
2024-12-26 18:33:50,280 - INFO - CSV file saved to: detailed_network_data_20241226_182014.csv

Conversion completed successfully!
Output file: detailed network data 20241226 182014.csv
```

Gambar 4.30 Data Selesai di Ekstrak

Setelah proses ekstraksi selesai maka akan mendapatkan *file* .csv yang berisi data ekstraksi data yang di-*capture* dari aplikasi Wireshark.

time	time_epoch	length	capture_length	highest_layer
Oct 31, 2024 14:24:35.530826000 SE Asia Standard Time	1730359476	75	75	DATA
Oct 31, 2024 14:24:35.546718000 SE Asia Standard Time	1730359476	86	86	TCP
Oct 31, 2024 14:24:35.602257000 SE Asia Standard Time	1730359476	524	524	TLS
Oct 31, 2024 14:24:35.602624000 SE Asia Standard Time	1730359476	3315	3315	TLS
Oct 31, 2024 14:24:35.602844000 SE Asia Standard Time	1730359476	85	85	TLS
Oct 31, 2024 14:24:35.615492000 SE Asia Standard Time	1730359476	56	56	TCP
Oct 31, 2024 14:24:35.615719000 SE Asia Standard Time	1730359476	56	56	TCP
Oct 31, 2024 14:24:35.615719000 SE Asia Standard Time	1730359476	56	56	TCP
Oct 31, 2024 14:24:35.615719000 SE Asia Standard Time	1730359476	56	56	TCP
Oct 31, 2024 14:24:35.714769000 SE Asia Standard Time	1730359476	395	395	TLS

Gambar 4.31 Cuplikan Data Hasil Ekstraksi

Kemudian data tersebut akan di standarisasi dengan Standarisasi *Scale s*ehingga menghasilkan data seperti pada Gambar 4.32.

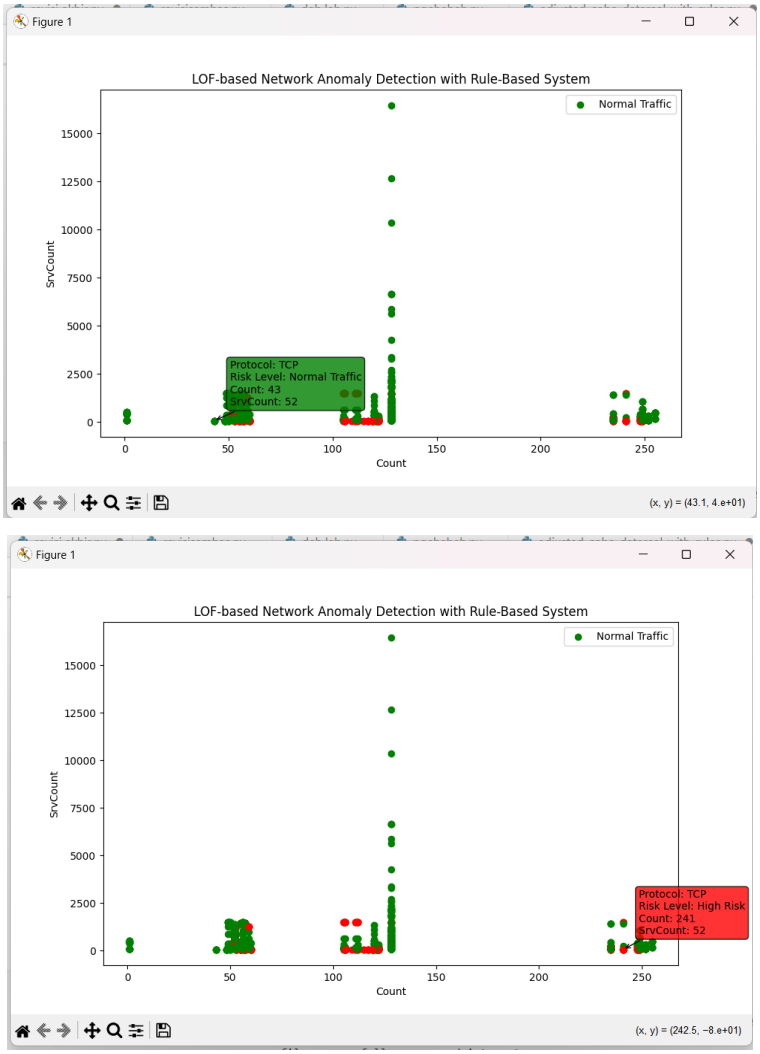
time_epoch	length	capture_length	eth_dst_oui
-5,034428233	-1,771719569	-1,771719569	-2,120024881
-5,019095592	-1,750511461	-1,750511461	0,471692577
-4,965510593	-0,906043155	-0,906043155	-2,120024881
-4,965156577	4,47503228	4,47503228	-2,120024881
-4,96494426	-1,752439471	-1,752439471	-2,120024881
-4,952741175	-1,808351756	-1,808351756	0,471692577
-4,952522187	-1,808351756	-1,808351756	0,471692577
-4,952522187	-1,808351756	-1,808351756	0,471692577
-4,952522187	-1,808351756	-1,808351756	0,471692577
-4,856957427	-1,154756423	-1,154756423	0,471692577

Gambar 4.32 Hasil Standarisasi

Selanjutnya, program akan di-*run* sehingga menampilkan hasil deteksi yang dilakukan oleh algoritma LOF dan *Rule-based System*.

```
2024-12-26 18:48:50,515 - INFO - Reading .pcap file from D:\datareal31okt14.25-14.35.pcapng
2024-12-26 18:51:44,289 - INFO - .pcap file successfully processed into DataFrame.
2024-12-26 18:51:44,301 - INFO - Selected numerical features for anomaly detection.
2024-12-26 18:51:44,305 - INFO - Data standardized.
2024-12-26 18:52:31,360 - INFO - Applied Local Outlier Factor for anomaly detection.
2024-12-26 18:52:31,363 - INFO - Found 1073 anomalies.
2024-12-26 18:52:34,145 - INFO - Displaying the interactive plot.
```

Gambar 4.33 *Log* Sistem Ketika di-*run*



Gambar 4.34 Hasil Deteksi Data *Real*

Berdasarkan hasil deteksi pada Gambar 4.34, dapat dilihat bahwa lalu lintas jaringan pada data *real* tersebut aman dari anomali serangan jaringan. Walaupun demikian, terlihat beberapa titik merah yang menandakan bahwa adanya anomali yang terdeteksi tetapi titik merah tersebut tidak dianggap membahayakan setelah melalui klasifikasi yang dilakukan oleh algoritma *Rule-based System*. Hal ini dibuktikan dengan nilai *srvcount* yang tergolong rendah, yang artinya kepadatan anomali jaringan tersebut masih dalam lingkup kepadatan lokal jaringan. Dengan demikian, ini membuktikan bahwa kombinasi dari algoritma LOF dan *Rule-based System* mampu mendeteksi anomali serangan jaringan dan mengklasifikasikannya terhadap level bahayanya.

BAB 5

PENUTUP

5.1. Kesimpulan

Melalui deteksi dini serangan jaringan melalui kombinasi *Local Outlier Factor* (LOF) dan *Rule-based System* yang telah dilakukan dapat ditarik beberapa kesimpulan seperti berikut:

1. Penelitian menunjukkan bahwa kombinasi algoritma LOF dan *Rule-based System* dapat digunakan secara efektif untuk mendeteksi dan mengklasifikasikan anomali dalam jaringan.
2. LOF memungkinkan deteksi anomali berdasarkan kepadatan lokal data, efektif dalam mengidentifikasi titik data yang mencurigakan atau abnormal dibandingkan dengan tetangganya.
3. *Rule-based System* memberikan klasifikasi lebih lanjut terhadap jenis ancaman berdasarkan protokol dan karakteristik lalu lintas yang terdeteksi.
4. Sistem yang dibangun mampu mendeteksi berbagai jenis serangan seperti DoS, *Probing*, TCP *Flood*, dan UDP *Flood* dengan akurasi yang baik.
5. Sistem yang dibangun mampu membantu dalam mitigasi ancaman yang berpotensi merusak infrastruktur jaringan.
6. Sistem yang dibangun bergantung pada aturan yang ditetapkan sebelumnya, sehingga kemampuan deteksi terhadap pola serangan baru atau yang belum diantisipasi mungkin terbatas.

5.2. Saran

Untuk pengembangan selanjutnya, disarankan agar sistem ini diintegrasikan dengan teknologi pembelajaran mesin yang lebih adaptif, seperti *deep learning*, yang memiliki kemampuan untuk belajar dan mendeteksi pola serangan baru tanpa harus mengandalkan aturan yang telah ditetapkan sebelumnya. Penggunaan model-model yang lebih canggih dapat meningkatkan kemampuan deteksi secara signifikan dan memungkinkan respons yang lebih cepat terhadap ancaman yang belum dikenal. Selain itu, penerapan mekanisme pencegahan otomatis, seperti integrasi dengan *firewall* atau sistem mitigasi serangan, dapat membantu menghentikan serangan secara langsung setelah terdeteksi, meningkatkan tingkat keamanan jaringan secara keseluruhan.

Selain itu, evaluasi dan pengujian lebih lanjut terhadap *dataset* yang lebih beragam dan kompleks sangat disarankan untuk memastikan bahwa sistem dapat berfungsi dengan baik dalam berbagai kondisi jaringan. Hal ini termasuk pengujian terhadap serangan yang berskala besar seperti *Distributed Denial of Service* (DDoS) yang sering menjadi ancaman bagi banyak sistem jaringan modern. Penelitian lanjutan juga perlu mempertimbangkan faktor-faktor seperti kinerja komputasi dan efisiensi waktu untuk memastikan bahwa sistem dapat beroperasi secara *real-time* tanpa mengganggu kinerja jaringan secara keseluruhan.

DAFTAR PUSTAKA

[1] J. Bughin, T. Kretschmer, and N. Van Zeebroeck, 'Digital Technology Adoption Drives Strategic Renewal for Successful Digital Transformation', *IEEE Eng. Manag. Rev.*, vol. 49, no. 3, pp. 103–108, 2021, doi: 10.1109/EMR.2021.3098663.

[2] S. Curtis, 'Digital transformation—the silver bullet to public service improvement?', *Public Money Manag.*, vol. 39, no. 5, pp. 322–324, Jul. 2019, doi: 10.1080/09540962.2019.1611233.

[3] A. Alvarenga, F. Matos, R. Godina, and J. C. O. Matias, 'Digital Transformation and Knowledge Management in the Public Sector', *Sustainability*, vol. 12, no. 14, p. 5824, Jul. 2020, doi: 10.3390/su12145824.

[4] T. M. Yuliandari, A. Putri, and Y. Rosmansyah, 'Digital Transformation in Secondary Schools: A Systematic Literature Review', *IEEE Access*, vol. 11, no. July, pp. 90459–90476, 2023, doi: 10.1109/ACCESS.2023.3306603.

[5] E. H. Budiarto, A. Erna Permanasari, and S. Fauziati, 'Unsupervised anomaly detection using K-Means, local outlier factor and one class SVM', *Proc. - 2019 5th Int. Conf. Sci. Technol. ICST 2019*, 2019, doi: 10.1109/ICST47872.2019.9166366.

[6] Z. Zhu, T. Song, J. Huang, and X. Zhong, 'Executive Cognitive Structure, Digital Policy, and Firms' Digital Transformation', *IEEE Trans. Eng. Manag.*, vol. 71, pp. 2579–2592, 2024, doi: 10.1109/TEM.2022.3190889.

[7] S. Trabelsi, 'Monitoring leaked confidential data', *2019 10th IFIP Int. Conf. New Technol. Mobil. Secur. NTMS 2019 - Proc. Work.*, pp. 1–5, 2019, doi: 10.1109/NTMS.2019.8763811.

[8] T. T. Lai, T. P. Tran, J. Cho, and M. Yoo, 'DoS attack detection using online learning techniques in wireless sensor networks', *Alexandria Eng. J.*, vol. 85, no. May, pp. 307–319, 2023, doi: 10.1016/j.aej.2023.11.022.

[9] A. A. Ojugo and R. E. Yoro, 'Forging a deep learning neural network intrusion detection framework to curb the distributed denial of service attack', *Int. J. Electr. Comput. Eng.*, vol. 11, no. 2, pp. 1498–1509, 2021, doi: 10.11591/ijece.v11i2.pp1498-1509.

[10] V. D. M. Rios, P. R. M. Inacio, D. Magoni, and M. M. Freire, 'Detection and Mitigation of Low-Rate Denial-of-Service Attacks: A Survey', *IEEE Access*, vol. 10, no. July, pp. 76648–76668, 2022, doi: 10.1109/ACCESS.2022.3191430.

[11] M. Li, Y. Yang, W. He, S. K. Mathew, V. De, and M. Seok, 'A Fully-Digital Variation-Tolerant Runtime Detector for PCB-Level Probing Attack in a 28-nm CMOS', *IEEE Solid-State Circuits Lett.*, vol. 6, pp. 245–248, 2023, doi: 10.1109/LSSC.2023.3310266.

[12] E. Tufan, C. Tezcan, and C. Acarturk, 'Anomaly-Based Intrusion Detection by

Machine Learning: A Case Study on Probing Attacks to an Institutional Network', *IEEE Access*, vol. 9, pp. 50078–50092, 2021, doi: 10.1109/ACCESS.2021.3068961.

[13] B. N. Ramkumar and T. Subbulakshmi, 'Tcp Syn Flood Attack Detection and Prevention System using Adaptive Thresholding Method', *ITM Web Conf.*, vol. 37, p. 01016, Mar. 2021, doi: 10.1051/itmconf/20213701016.

[14] A. Bijalwan, M. Wazid, E. S. Pilli, and R. C. Joshi, 'Forensics of Random-UDP Flooding Attacks', *J. Networks*, vol. 10, no. 5, May 2015, doi: 10.4304/jnw.10.5.287-293.

[15] T. Sui *et al.*, 'A Real-Time Hidden Anomaly Detection of Correlated Data in Wireless Networks', *IEEE Access*, vol. 8, pp. 60990–60999, 2020, doi: 10.1109/ACCESS.2020.2984276.

[16] B. Min, J. Yoo, S. Kim, D. Shin, and D. Shin, 'Network Anomaly Detection Using Memory-Augmented Deep Autoencoder', *IEEE Access*, vol. 9, pp. 104695–104706, 2021, doi: 10.1109/ACCESS.2021.3100087.

[17] S. Kamamura, Y. Takei, M. Nishiguchi, Y. Hayashi, and T. Fujiwara, 'Network Anomaly Detection Through IP Traffic Analysis With Variable Granularity', *IEEE Access*, vol. 11, no. October, pp. 129818–129828, 2023, doi: 10.1109/ACCESS.2023.3334212.

[18] H. Kye, M. Kim, and M. Kwon, 'Hierarchical Detection of Network Anomalies : A Self-Supervised Learning Approach', *IEEE Signal Process. Lett.*, vol. 29, pp. 1908–1912, 2022, doi: 10.1109/LSP.2022.3203296.

[19] J. Auskalnis, N. Paulauskas, and A. Baskys, 'Application of Local Outlier Factor algorithm to detect anomalies in computer network', *Elektron. ir Elektrotechnika*, vol. 24, no. 3, pp. 96–99, 2018, doi: 10.5755/j01.eie.24.3.20972.

[20] V. Chandola, A. Banerjee, and V. Kumar, 'Anomaly Detection: A Survey', *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, Jul. 2009, doi: 10.1145/1541880.1541882.

[21] C. Grosan and A. Abraham, 'Rule-Based Expert Systems', in *Intelligent Systems Reference Library*, vol. 17, 2011, pp. 149–185. doi: 10.1007/978-3-642-21004-4_7.

[22] M. A. Setitra, M. Fan, I. Benkhaddra, and Z. E. A. Bensalem, 'DoS/DDoS attacks in Software Defined Networks: Current situation, challenges and future directions', *Comput. Commun.*, vol. 222, no. November 2023, pp. 77–96, Jun. 2024, doi: 10.1016/j.comcom.2024.04.035.

[23] A. Gramfort *et al.*, 'StandardScaler'. https://github.com/scikit-learn/scikit-learn/blob/6e9039160/sklearn/preprocessing/_data.py#L696

LAMPIRAN

Lampiran 1. Glosarium

Autoencoder : jenis jaringan saraf buatan yang digunakan untuk merepresentasikan data dalam bentuk yang lebih sederhana (*encoding*) dan kemudian merekonstruksinya kembali (*decoding*) dengan tingkat kesalahan yang minimal.

Count : frekuensi kemunculan suatu nilai atau kategori dalam *dataset*.

Dstbytes (*destination bytes*) : jumlah *byte* data yang dikirim dari sumber ke tujuan dalam sebuah koneksi jaringan atau komunikasi tertentu.

Dsthostserrorrate (*destination host's error rate*) : rasio atau tingkat kesalahan yang terjadi pada *host* tujuan selama komunikasi jaringan.

Flood Attack : jenis serangan siber di mana penyerang membanjiri target dengan lalu lintas data yang sangat tinggi untuk membuatnya tidak dapat berfungsi atau merespons permintaan yang sah.

k-*nearest neighbors* (k-NN) : algoritma pembelajaran mesin yang digunakan untuk klasifikasi dan regresi.

Local Reachability Density (LRD) : metrik yang digunakan dalam analisis kepadatan lokal data.

Ping sweep : teknik pemindaian jaringan untuk menentukan host mana yang aktif dalam sebuah jaringan.

Port scanning : teknik untuk memeriksa port yang terbuka atau tertutup di komputer atau perangkat jaringan.

Srcbytes (*source bytes*) : jumlah *byte* data yang dikirim dari sumber (*source*) ke tujuan (*destination*) dalam komunikasi jaringan.

Srvcount (*service count*) : jumlah layanan tertentu yang diakses oleh sebuah koneksi atau *host* dalam periode waktu tertentu.

Srvserrorrate (*service error rate*) : rasio kesalahan yang terjadi pada layanan tertentu selama komunikasi jaringan.

Vulnerability scanning : proses otomatis untuk mengidentifikasi celah keamanan atau kerentanan dalam sistem, aplikasi, atau jaringan.

Lampiran 2. *Source Code*

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import LocalOutlierFactor
import matplotlib.pyplot as plt

file_path1 =
r"C:\Users\riski\Downloads\refined_network_anomaly_data_updated_7_1.csv"
file_path2 =
r"C:\Users\riski\Downloads\refined_train_dataset_corrected_attack_type_2
1_1.csv"

data1 = pd.read_csv(file_path1, low_memory=False)
data2 = pd.read_csv(file_path2, low_memory=False)

combined_data = pd.concat([data1, data2], ignore_index=True)

numerical_features = combined_data.select_dtypes(include=[np.number])

scaler = StandardScaler()
scaled_data = scaler.fit_transform(numerical_features)

lof = LocalOutlierFactor(n_neighbors=20, contamination=0.05)
combined_data['anomaly'] = lof.fit_predict(scaled_data)

anomalies = combined_data[combined_data['anomaly'] == -1]

def classify_risk(protocol):
    risk_levels = {
        'tcp': ('TCP Attack', 'red'),
        'udp': ('UDP Attack', 'orange'),
        'dos': ('DoS Attack', 'yellow'),
        'probing': ('Probing Attack', 'red'),
        'normal': ('Normal Traffic', 'green'),
        'unknown': ('Unknown Attack', 'black')
    }
    return risk_levels.get(protocol, ('Unknown Attack', 'black'))

def rule_based_classification(row):
    if row['protocoltype'] == 'tcp' and row['srcbytes'] > 1000:
        return 'High Risk', 'red'
```

```
    elif row['protocoltype'] == 'udp' and row['dstbytes'] > 1500:
        return 'Medium Risk', 'orange'
    elif row['protocoltype'] == 'dos' and row['count'] > 50:
        return 'High Risk', 'red'
    elif row['protocoltype'] == 'probing' and (row['srvcount'] < 50 or
row['count'] > 20):
        return 'High Risk', 'red'
    elif row['protocoltype'] == 'normal':
        return 'Normal Traffic', 'green'
    else:
        return classify_risk('unknown')

fig, ax = plt.subplots(figsize=(8, 6))

scatters = []
for index, row in combined_data.iterrows():
    risk, color = rule_based_classification(row)
    scatter = ax.scatter(index, row['srcbytes'], color=color)
    scatters.append((scatter, row))

ax.set_title('LOF-based Network Anomaly Detection with Rule-Based
System')
ax.set_xlabel('Index (Observation Points)')
ax.set_ylabel('Source Bytes (Traffic Volume)')

annot = ax.annotate("", xy=(0, 0), xytext=(15, 15), textcoords="offset
points",
                    bbox=dict(boxstyle="round", fc="w"),
                    arrowprops=dict(arrowstyle="->"))
annot.set_visible(False)

def update_annot(ind):
    pos = scatters[ind][0].get_offsets()[0]
    annot.xy = pos
    row = scatters[ind][1]
    risk, _ = rule_based_classification(row)
    text = f"Protocol: {row['protocoltype']}\nRisk Level:
{risk}\nSrcBytes: {row['srcbytes']}"
    annot.set_text(text)
    annot.get_bbox_patch().set_facecolor(scatters[ind][0].get_facecolor(
)[0])
    annot.get_bbox_patch().set_alpha(0.6)

def hover(event):
    vis = annot.get_visible()
    if event.inaxes == ax:
        for i, (scatter, row) in enumerate(scatters):
            cont, _ = scatter.contains(event)
```

```
            if cont:
                update_annot(i)
                annot.set_visible(True)
                fig.canvas.draw_idle()
                return
    if vis:
        annot.set_visible(False)
        fig.canvas.draw_idle()

fig.canvas.mpl_connect("motion_notify_event", hover)

plt.show()
```