BAB II

TINJAUAN PUSTAKA

2.1 Justifikasi Solusi

Sebelum melakukan penelitian, dilakukan riset terhadap penelitian yang sudah dilakukan sebelumnya sebagai acuan.

2.1.1 *SSH and FTP brute-force Attacks Detection in Computer Network: LSTM and Machine Learning Approaches*

Penelitian yang Md Delwar Hossain, et al. bertujuan untuk mengembangkan metode deteksi serangan *brute force* pada protokol SSH dan FTP menggunakan pendekatan *deep learning* dengan algoritma *Long Short-Term Memory (LSTM)* [12]. Penelitian ini membandingkan performa LSTM dengan beberapa algoritma pembelajaran mesin klasik yaitu *J48*, *Naïve Bayes*, *Decision Table*, *Random Forest*, *dan k-Nearest Neighbor* untuk tujuan deteksi tambahan.

Penelitian ini menggunakan dataset CIC-IDS-2017 yang telah berlabel dengan komposisi 432.074 trafik normal berbanding dengan 13.835 trafik serangan (7.938 trafik FTP dan 5.897 trafik SSH). Evaluasi efektivitas dilakukan dengan membandingkan performa antara model LSTM dan algoritma pembalajaran mesin tradisional menggunakan rasio pemisahan 80:20 untuk pelatihan dan validasi. Hasil penelitian menunjukkan bahwa model LSTM mencapai akurasi 99.88% dengan metrik ROC-AUC sebesar 1.00.

Namun demikian, penelitian ini memiliki beberapa keterbatasan metodologi yang siginifikan, seperti:

1. Ketidakseimbangan kelas yang ekstrim menyebabkan skor akurasi yang berlebihan, dimana penulis mengakui bahwa, "*due to the fact that benign class elements are high in the dataset, the detection accuracy with the proposed deep learning classifiers is also high*".
2. Penelitian hanya menggunakan satu sumber dataset saja.

3. Tidak ada implementasi atau validasi dalam lingkungan nyata untuk menguji kemampuan model.
4. Penelitian tidak mempertimbangkan varian serangan lain seperti *slow-rate*.

Dari penelitian tersebut, hal yang dapat dijadikan acuan yaitu:

1. Penggunaan arsitektur LSTM terbukti efektif untuk serangan *brute force* pada protokol SSH dengan metrik akurasi tinggi, sehingga LSTM menjadi referensi algoritma yang digunakan dalam penelitian penulis.
2. Keterbatasan penelitian terdahulu dalam hal ketidakseimbangan dataset dan kurangnya implementasi nyata menjadi *gap* yang dapat diisi oleh penelitian ini.

2.1.2 *HTTP Low and Slow DoS Attack Detection using LSTM*

Penelitian yang dilakukan oleh Gogoi dan Ahmed mengkaji efektivitas model *Long Short-Term Memory* (LSTM) dalam mendeteksi serangan HTTP *low and slow Denial of Service* (DoS), yaitu serangan yang mengirimkan permintaan HTTP secara sangat lambat untuk menghabiskan sumber daya server tanpa melakukan *flooding* [13]. Serangan jenis ini menargetkan aplikasi atau komponen spesifik pada server dan memanfaatkan sifat server berbasis *thread*, seperti Apache dan IIS, yang akan mempertahankan koneksi hingga permintaan selesai. Karena tidak menghasilkan lalu lintas dalam jumlah besar, serangan *low and slow* sulit dideteksi oleh mekanisme tradisional jaringan.

Penelitian ini menggunakan dataset CIC-DoS serta dataset sintetis yang dibangun khusus untuk menyerupai pola serangan *slow-rate* pada protokol HTTP. Model LSTM dilatih untuk mempelajari dinamika temporal dari koneksi HTTP dan membedakan trafik lambat yang berbahaya dari permintaan normal. Hasil evaluasi menunjukkan bahwa model LSTM mampu mencapai akurasi 99 persen, mengartikan bahwa pendekatan pembelajaran berbasis urutan ini dapat menangkap pola temporal halus yang tidak terdeteksi oleh alat deteksi berbasis aturan.

Dari penelitian tersebut, hal yang dapat dijadikan acuan adalah:

1. LSTM efektif untuk mendeteksi serangan *slow-rate* yang berlangsung lambat dan bersifat temporal karena mampu mempelajari perubahan perilaku koneksi sepanjang waktu.
2. Serangan *low and slow* memerlukan pendekatan deteksi yang melihat urutan peristiwa, bukan hanya volume atau frekuensi, sehingga LSTM relevan untuk jenis serangan ini.

Penelitian ini dijadikan dasar untuk penggunaan model LSTM dalam mendeteksi serangan *slow-rate* pada protokol SSH, mengigat karakteristik temporalnya serupa dan sama-sama tidak menimbulkan lonjakan lalu lintas yang mudah diamati oleh detektor berbasis aturan.

2.1.3 *On the Detection Capabilities of Signature-Based Intrusion Detection System in the Context of Web Attacks*

Penelitian yang dilakukan oleh oleh J. Díaz-Verdejo, et al. mengevaluasi kemampuan deteksi sistem intrusi berbasis *signature* dalam konteks serangan *web*, menyoroti keterbatasan fundamental pendekatan berbasis aturan dalam mengidentifikasi ancaman baru dan kompleks [14]. Studi ini menunjukkan bahwa meskipun sistem berbasis *signature* efektif untuk serangan yang telah dikenal, sistem tersebut sering gagal mendeteksi serangan dengan pola yang belum terdefinisi atau serangan yang dirancang untuk mengelabui deteksi berbasis aturan.

Penelitian ini mengidentifikasi beberapa kelemahan kritis sistem berbasis aturan: ketidakmampuan beradaptasi dengan varian serangan baru, ketergantungan pada ambang batas statis yang dapat dikelabui oleh penyerang, dan keterbatasan dalam menganalisis pola temporal yang kompleks. Hasil eksperimen menunjukkan bahwa sistem *signature-based* memiliki tingkat *false negative* yang tinggi terhadap serangan yang menggunakan teknik *evasion* seperti *slow-rate*.

Dari penelitian tersebut, hal yang dapat dijadikan acuan adalah:

1. Keterbatasan sistem berbasis aturan dalam mendeteksi serangan adaptif yang menghindar dari ambang batas.

2. Kebutuhan pendekatan yang dapat mempelajari pola dinamis dari sebuah serangan.
3. Pentingnya analisis perilaku alamat IP dalam mengidentifikasi serangan, dibandingkan dengan penyamaan pola serangan yang statis.

2.1.4 *Transferability of Machine Learning Models Learned from Public Intrusion Detection Datasets: The CICIDS2017 Case Study*

Penelitian yang dilakukan oleh Marta Catillo, et al. menginvestigasi transferabilitas model pembelajaran mesin yang dilatih menggunakan dataset CICIDS2017. Penelitian ini menguji model dengan dataset yang tidak pernah dilihat sebelumnya meskipun memiliki karakteristik serupa [15]. Hasil menunjukkan bahwa akurasi deteksi yang mendekati sempurna pada dataset publik tidak selalu mentransfer ke praktik nyata.

Penelitian ini menggunakan metodologi yang membandingkan performa model pada dataset pelatihan orisinil dengan dataset pengujian yang indendependen namun mirip. Eksperimen menunjukkan degradasi performa yang signifikan ketika model diaplikasikan pada data baru, mengindikasikan bahwa model sangat bergantung atau menghafal karakteristik serangan sehingga menjadikan dataset pelatihan tidak representatif terhadap lingkungan produksi nyata.

Dari penelitian tersebut, hal yang dapat dijadikan acuan adalah:

1. Pentingnya validasi model pada lingkungan uji coba yang nyata.
2. Performa tinggi pada dataset tidak menjamin efektivitas model, peneliti menyarankan untuk menggabungkan atau menguji silang antar dataset setelah model dilatih.

2.1.5 *Evaluation of Cluster Based Anomaly Detection*

Penelitian yang dilakukan oleh Ajay Sreenivasulu mengevaluasi penggunaan pendekatan *clustering-based anomaly detection* dengan fokus pada algoritma *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) untuk mendeteksi anomali pada dataset yang tidak berlabel [16]. Penelitian ini bertujuan untuk menilai sejauh mana algoritma *clustering*, khususnya DBSCAN, mampu membedakan data normal dan data anomali tanpa memerlukan proses pelabelan sebelumnya. Evaluasi performa dilakukan menggunakan metrik *precision*, *recall*, dan *F1-score* terhadap *ground truth* yang telah diketahui sebelumnya.

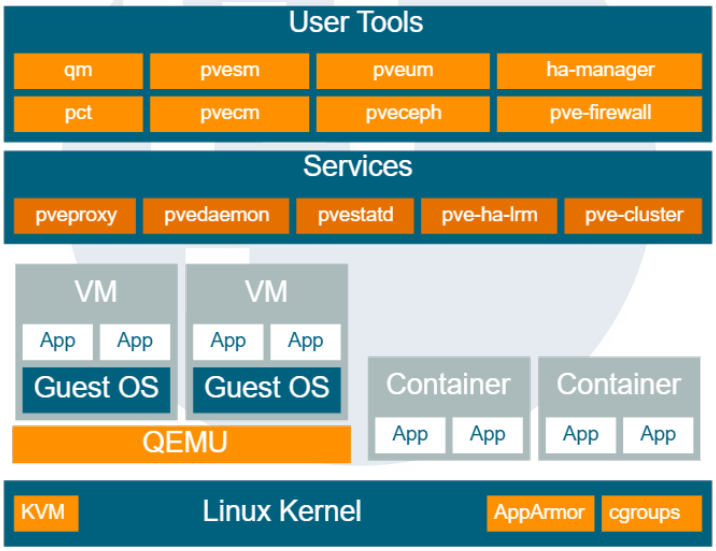
Hasil penelitian menunjukkan bahwa DBSCAN memiliki keunggulan dalam mengidentifikasi anomali pada data dengan kepadatan yang tidak seragam dan mampu menangani bentuk *cluster* yang kompleks.

Dari penelitian tersebut, hal yang dapat dijadikan acuan dalam penelitian ini adalah:

1. Pendekatan *clustering-based anomaly detection* terbukti dapat digunakan sebagai metode deteksi awal pada data yang tidak berlabel, khususnya dalam mengidentifikasi pola yang menyimpang dari mayoritas data.
2. DBSCAN efektif dalam mendeteksi anomali berbasis kepadatan, namun keberadaan *noise* yang tidak terklasifikasi menunjukkan bahwa *clustering* saja belum cukup untuk membedakan seluruh perilaku anomali secara akurat.
3. Keterbatasan DBSCAN dalam menangkap dinamika temporal dan ketergantungannya terhadap parameter membuka peluang untuk dikombinasikan dengan model pembelajaran berbasis urutan, seperti LSTM, guna meningkatkan akurasi deteksi serangan *slow-rate* dan serangan adaptif lainnya.

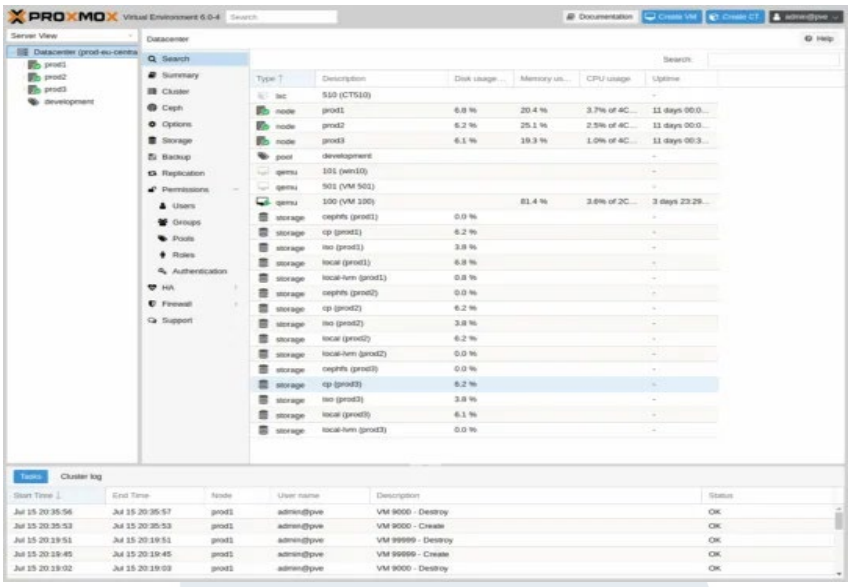
2.2 Virtualisasi Proxmox

Proxmox Virtual Environment (Proxmox VE atau PVE) adalah sebuah platform manajemen virtualisasi server berbasis *open-source* yang dirancang untuk mengelola infrastruktur teknologi secara terpusat [2]. Proxmox berfungsi sebagai sistem operasi *host* yang memungkinkan satu server fisik menjalankan beberapa sistem operasi virtual secara simultan. Pendekatan ini meningkatkan efisiensi pemanfaatan sumber daya perangkat keras karena satu mesin fisik dapat digunakan untuk menjalankan berbagai layanan secara terisolasi dalam lingkungan virtual yang berbeda.



Gambar 2.1 Arsitektur Internal Proxmox.
Sumber: [17]

Proxmox mengintegrasikan dua teknologi virtualisasi utama, yaitu *Kernel-based Virtual Machine* (KVM) dan *Linux Containers* (LXC). KVM memungkinkan pembuatan mesin virtual yang sepenuhnya terisolasi, di mana setiap VM memiliki kernal dan sistem operasi sendiri layaknya komputer fisik yang berdiri sendiri. Sementara itu, LXC merupakan teknologi virtualisasi berbasis kontainer yang lebih ringan, di mana beberapa kontainer berbagi kernel dengan sistem operasi *host* Proxmox, namun tetap memiliki ruang pengguna yang terpisah dan terisolasi. Kombinasi kedua teknologi ini memungkinkan fleksibilitas dalam pengelolaan beban kerja sesuai kebutuhan sistem.



Gambar 2.2 Tampilan Web Antarmuka Proxmox

Selain itu, Proxmox menyediakan antarmuka berbasis web yang memudahkan administrator dalam melaukan konfigurasi, pemantauan, dan manajemen mesin virtual maupun kontainer secara terpusat tanpa ketergantungan penuh pada antarmuka baris perintah. Fitur ini mendukung pengelolaan infrastruktur server secara efisien dan terstruktur.

Dalam penelitian ini, Proxmox digunakan sebagai sistem operasi server yang menjadi sumber utama pengambilan data log autentikasi sekaligus sebagai lingkungan pengujian sistem deteksi intrusi. Karena Proxmox berbasis sistem operasi Linux, seluruh aktivitas autentikasi, termasuk akses melalui protokol *Secure Shell* (SSH), dicatat pada berkas log autentikasi yang tersimpan pada direktori $/var/log/auth.log$. Log autentikasi inilah yang akan diambil dan digunakan sebagai dataset penelitian untuk menganalisis pola serangan.

2.3 Protokol *Secure Shell* (SSH)

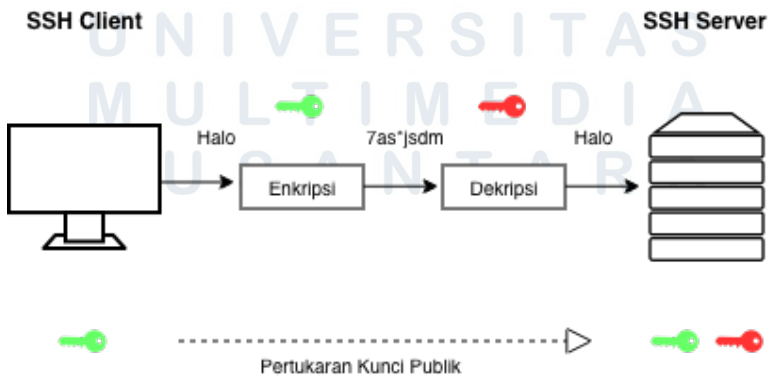
Secure Shell (SSH) merupakan protokol jaringan kriptografi yang dirancang untuk menyediakan mekanisme komunikasi jarak jauh yang aman melalui jaringan yang tidak terpercaya. Protokol ini dikembangkan untuk menggantikan protokol akses jarak jauh konvensional seperti Telnet, rlogin, dan rcp yang mentransmisikan data dalam bentuk teks biasa tanpa perlindungan keamanan [18]. SSH pertama kali diperkenalkan oleh Tatu Ylönen dari Helsinki University of Technology pada tahun 1995 sebagai respons terhadap insiden

pencurian kata sandi (*password sniffing*) yang terjadi di jaringan universitas tempat beliau bekerja [19].

SSH dipublikasikan secara resmi pada Juli 1995 dan dengan cepat diadopsi secara luas, dengan estimasi lebih dari 20.000 pengguna di lebih dari 50 negara pada akhir tahun yang sama [20]. Seiring perkembangannya, protokol ini mengalami peningkatan signifikan dari versi awal SSH-1 menuju SSH-2, yang diperkenalkan pada tahun 1996 untuk mengatasi berbagai kelemahan keamanan yang ditemukan pada versi sebelumnya [21]. SSH-2 kemudian distandarisasi oleh Internet Engineering Task Force (IETF) melalui serangkaian *Request for Comments* (RFC), yaitu RFC 4250 hingga RFC 4255, yang mendefinisikan arsitektur protokol, lapisan *transport*, mekanisme autentikasi, dan protokol koneksi SSH [22].

Secara arsitektur, SSH menggunakan tiga lapisan sebagai komponen utama [23], yaitu:

1. Protokol Autentikasi SSH (RFC 4252), menangani autentikasi seperti kata sandi, kunci publik, dan autentikasi berbasis *host*.
2. Protokol SSH Transport Layer (RFC 4253) bertanggung jawab untuk pembentukan saluran yang terenkripsi, mengandung algoritma *Message Authentication Code* (MAC) dan algoritma kompresi untuk melindugi integritas data yang ditransmisikan.
3. Protokol Koneksi SSH (RFC 4254) mengelola layanan SSH setelah saluran aman dan autentikasi berhasil dibentuk, termasuk eksekusi program jarak jauh, akses *shell*, dan *port forwarding*.



Gambar 2.3 Alur Kerja Protokol SSH

Proses koneksi SSH diawali dengan pembentukan koneksi *Transmission Control Protocol* (TCP), dilanjutkan dengan pertukaran informasi versi dan negosiasi algoritma kriptografi. Selanjutnya dilakukan pertukaran kunci menggunakan algoritma kriptografi asimetris, umumnya berbasis Diffie–Hellman, untuk menghasilkan kunci sesi. Setelah kunci sesi terbentuk, proses autentikasi pengguna dilakukan sebelum sesi SSH aktif sepenuhnya dan layanan koneksi dapat digunakan [21]. Dalam operasionalnya, SSH memanfaatkan kombinasi kriptografi simetris untuk enkripsi data, kriptografi asimetris untuk pertukaran kunci, serta fungsi hash kriptografi untuk menjamin integritas data. Setiap sesi SSH menggunakan kunci sesi yang berbeda, sehingga kompromi terhadap satu sesi tidak berdampak pada sesi komunikasi lainnya.

SSH telah menjadi standar untuk akses jarak jauh yang aman dan digunakan secara luas dalam pengelolaan sistem berbasis Unix dan Linux. Diperkirakan lebih dari 95% server yang terhubung ke internet menggunakan SSH sebagai mekanisme utama untuk administrasi jarak jauh [24]. Tingginya tingkat adopsi ini dipengaruhi oleh kemampuannya dalam menyediakan keamanan komunikasi yang kuat, serta ketersediaan implementasi *open-source* seperti OpenSSH yang diperkenalkan pada tahun 1999 dan menjadi implementasi SSH paling umum digunakan hingga saat ini [25].

Dalam konteks sistem operasi Linux, termasuk Proxmox Virtual Environment yang digunakan pada penelitian ini, seluruh aktivitas autentikasi SSH secara otomatis dicatat dalam berkas log autentikasi yang tersimpan pada direktori $/var/log/auth.log$. Berkas log ini mencatat setiap percobaan koneksi SSH beserta informasi penting seperti tanda waktu (*timestamp*), alamat IP sumber, nama pengguna yang digunakan, serta status autentikasi berhasil atau gagal. Karakteristik log autentikasi yang bersifat kronologis dan kaya akan informasi temporal menjadikannya sumber data yang relevan untuk analisis keamanan. Data ini memungkinkan rekonstruksi urutan kejadian serangan, identifikasi pola perilaku penyerang, serta ekstraksi fitur-fitur yang menjadi dasar dalam analisis dan deteksi serangan SSH pada tahap selanjutnya.

2.4 Serangan *Brute-Force*

Serangan *brute force* merupakan metode penyerangan terhadap mekanisme autentikasi yang dilakukan dengan melakukan *trial and error*, di mana penyerang secara sistematis mencoba berbagai kombinasi kata sandi atau kredensial hingga menemukan kombinasi yang pas. Menurut MITRE dalam skema *Common Attack Pattern Enumeration and Classification (CAPEC)*, serangan *brute force* didefinisikan sebagai upaya memperoleh akses ke aset yang dilindungi oleh kata sandi melalui eksplorasi menyeluruh atau sebagian besar ruang kemungkinan kata sandi, dengan harapan menemukan kata sandi yang benar atau secara fungsional setara. Efektivitas serangan ini sangat dipengaruhi oleh kompleksitas kata sandi, ukuran himpunan karakter, serta kapasitas komputasi yang dimiliki penyerang.

Kompleksitas waktu serangan *brute force* meningkat secara eksponensial seiring dengan bertambahnya panjang kata sandi atau kunci enkripsi. Untuk kata sandi dengan panjang k dan himpunan karakter berjumlah n, jumlah kemungkinan kombinasi yang harus diuji adalah n^k. Sebagai contoh, kata sandi sepanjang 8 karakter yang terdiri atas huruf besar, huruf kecil, dan angka memiliki lebih dari 218 triliun kemungkinan kombinasi. Meskipun jumlah ini sangat besar, perkembangan teknologi komputasi paralel telah secara siginifikan menurunkan waktu yang dibutuhkan untuk melakukan eksplorasi ruang pencarian tersebut. Pemanfaatan *Graphic Processing Unit* (GPU) dan *Field-Programmable Gate Array* (FPGA) memungkinkan penyerang melakukan jutaan hingga miliran percobaan dalam waktu singkat, terutama pada skema *hash* tertentu yang relatif cepat untuk dihitung [26].

Serangan *brute force* tetap menjadi metode yang populer karena sifatnya yang bersifat universal dan selalu dapat diterapkan dalam sistem autentikasi berbasis kata sandi. Metode ini tidak memerlukan eksploitasi kerentanan spesifik pada perangkat lunak, melainkan memanfaatkan kelemahan pada penggunaan kredensial, seperti kata sandi yang lemah atau digunakan ulang. Selain itu, kemudahan implementasi menjadikan *brute force* sebagai pilihan umum bagi penyerang, karena serangan dapat diotomasi menggunakan alat yang tersedia secara luas.

Serangan *brute force* pada autentikasi umumnya dapat dibedakan menjadi dua pendekatan utama, yaitu serangan *brute force* konvensional (serangan dengan frekuensi tinggi) dan serangan *brute force* dengan teknik *slow-rate* (lambat).

2.4.1 Serangan Frekuensi Tinggi

Serangan *brute force* dengan frekuensi tinggi, yang sering disebut sebagai *fast attack*, dilakukan dengan mengirimkan sejumlah besar percobaan login dalam waktu yang singkat. Pendekatan ini bertujuan untuk memperoleh akses secepat mungkin dengan memanfaatkan kemampuan otomatisasi dan kecepatan komputasi yang dimiliki penyerang. Dalam praktiknya, *fast attack* menghasilkan lonjakan aktivitas autentikasi gagal yang signifikan dalam jendela waktu yang pendek.

Pola serangan dengan frekuensi tinggi ini relatif mudah dikenali oleh sistem keamanan berbasis aturan, seperti mekanisme *rate limiting* atau pemblokiran otomatis berbasis ambang batas jumlah kegagalan login. Ketika jumlah percobaan autentikasi gagal melampaui ambang yang telah ditentukan dalam periode waktu tertentu, sistem pertahanan akan memicu peringatan atau melakukan pemblokiran terhadap alamat IP sumber serangan. Oleh karena itu, meskipun *fast attack* sederhana dan efektif pada sistem dengan pertahanan lemah, teknik ini cenderung kurang berhasil pada sistem yang telah dilengkapi dengan mekanisme deteksi dasar.

2.4.2 Serangan dengan Teknik *Slow-Rate*

Sebagai respon terhadap mekanisme pertahanan berbasis aturan tersebut, penyerang mengembangkan variasi *brute force* yang lebih adaptif, yaitu serangan *brute force* dengan teknik *slow-rate*. Teknik *slow-rate* merupakan pendekatan yang dirancang untuk menghindari deteksi dengan cara menyebarkan percobaan login dalam interval waktu yang panjang dan dengan volume yang sangat rendah. Konsep *low and slow* ini pertama kali diperkenalkan dalam konteks serangan *Distributed*

Denial of Service (DDoS) dan kemudian diadaptasi untuk serangan terhadap mekanisme autentikasi.

Berbeda dengan *fast attack*, serangan *slow-rate* tidak menunjukkan lonjakan aktivitas yang mencolok. Serangan ini biasanya dilakukan dengan interval waktu antar percobaan yang panjang, mulai dari beberapa menit hingga beberapa jam, serta melibatkan banyak alamat IP yang tersebar secara geografis. Penyerang sering memanfaatkan *botnet*, *Virtual Private Network* (VPN), *proxy*, atau jaringan *The Onion Router* (TOR) untuk melakukan rotasi alamat IP, sehingga setiap sumber hanya memberikan kontribusi serangan yang kecil dan tidak melewati ambang batas pemblokiran [7]. Selain itu, kamus kata sandi dapat didistribusikan di antara beberapa bot atau target untuk menghindari pola serangan yang mudah dikenali.

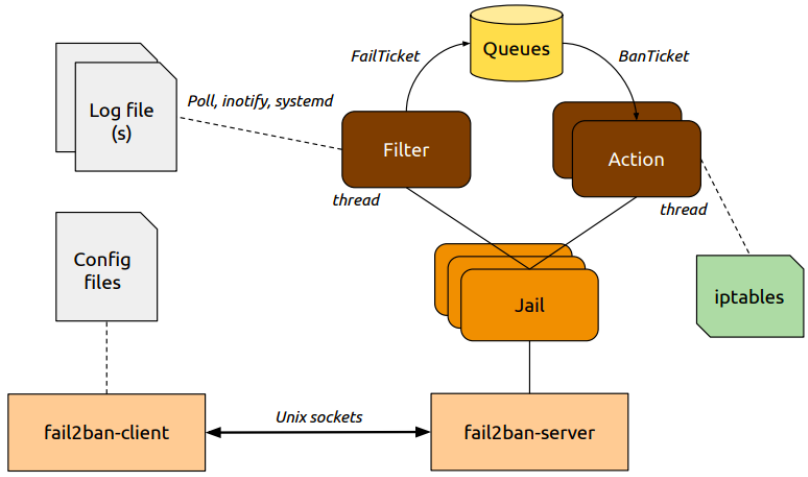
2.5 Deteksi Serangan Berbasis Aturan (*Rule Based*)

Sistem deteksi serangan berbasis aturan (*rule-based intrusion detection system*) merupakan pendekatan deteksi yang bekerja berdasarkan pengetahuan awal mengenai pola-pola serangan yang telah diketahui sebelumnya. Sistem ini melakukan proses pencocokan antara aktivitas yang diamati dengan aturan atau tanda (*signature*) yang telah didefinisikan sebelumnya untuk mengidentifikasi perilaku yang dianggap mencurigakan atau berbahaya [14]. Menurut Kumar dan Sangwan, sistem deteksi intrusi berbasis aturan melakukan pemantauan terhadap aktivitas jaringan atau sistem untuk mendeteksi pola perilaku tertentu dan menghasilkan peringatan ketika aktivitas tersebut sesuai dengan aturan yang telah ditetapkan [27].

Pendekatan berbasis aturan memiliki keunggulan utama berupa efisiensi komputasi yang tinggi serta tingkat *false positive* yang relatif rendah, karena deteksi hanya dilakukan terhadap pola yang telah terdefinisi secara eksplisit. Sistem ini sangat efektif dalam mendeteksi serangan yang memiliki karakteristik jelas dan telah dikenal, seperti serangan *brute force* dengan frekuensi tinggi. Namun, keterbatasan utama dari pendekatan ini adalah ketergantungannya pada aturan statis, sehingga sistem sulit mendeteksi

serangan baru (*zero-day attack*) atau serangan yang dirancang untuk tetap berada di bawah ambang batas aturan yang telah ditentukan.

Salah satu implementasi sistem deteksi berbasis aturan yang banyak digunakan pada lingkungan server berbasis Linux adalah Fail2Ban. Fail2Ban merupakan perangkat lunak pencegah intrusi yang bekerja dengan memantau berkas log dari berbagai layanan, seperti SSH, Apache, dan Postfix, untuk mendeteksi pola perilaku mencurigakan, khususnya percobaan autentikasi yang gagal secara berulang. Ketika pola tertentu terdeteksi, Fail2Ban secara otomatis menerapkan aturan *firewall* dinamis untuk memblokir alamat IP sumber serangan dalam jangka waktu tertentu.



Gambar 2.4 Alur Kerja Fail2Ban

Pada gambar 2.4, Fail2Ban bekerja dengan memantau berkas log sistem secara kontinu untuk mendeteksi pola aktivitas yang mencurigakan, seperti percobaan autentikasi yang gagal berulang kali. Log yang dihasilkan oleh layanan (misalnya SSH) diproses oleh filter berdasarkan aturan yang telah dikonfigurasi, kemudian kejadian yang terdeteksi dikirimkan ke dalam antrean (*queue*) untuk dievaluasi. Apabila jumlah atau pola kegagalan memenuhi kriteria yang ditentukan, Fail2Ban mengeksekusi aksi yang sesuai melalui mekanisme *jail*, umumnya dengan menambahkan aturan *firewall* secara dinamis menggunakan 'iptables' untuk memblokir alamat IP sumber dalam jangka waktu tertentu. Seluruh proses ini dikelola oleh 'fail2ban-server' dan

dapat dikontrol melalui ‘fail2ban-client’, sehingga memungkinkan penerapan perlindungan otomatis berbasis aturan terhadap serangan *brute force*.

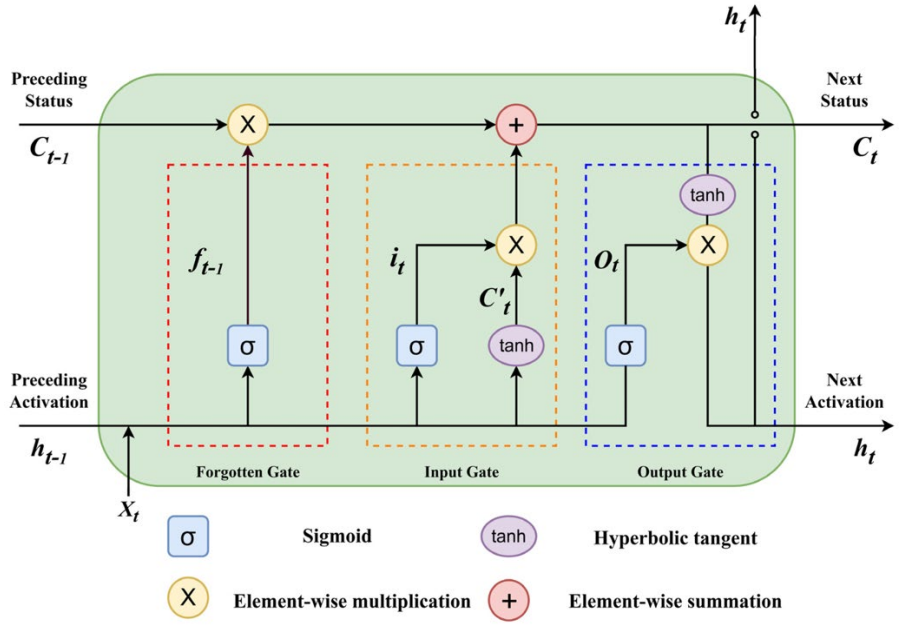
Mekanisme deteksi Fail2Ban yang sederhana dan berbasis log memungkinkan untuk memberikan perlindungan langsung terhadap serangan *brute force* tanpa memerlukan pemrosesan yang kompleks.

Namun, Fail2Ban sepenuhnya bergantung pada aturan ambang batas yang telah ditentukan, efektivitas Fail2Ban sangat dipengaruhi oleh konfigurasi aturan tersebut. Penyerang yang memahami pola kerja sistem ini dapat menyesuaikan strategi serangan agar tetap berada di bawah ambang batas deteksi. Selain itu, kebutuhan untuk menyesuaikan parameter aturan secara manual terhadap berbagai pola serangan menjadikan pendekatan berbasis aturan kurang fleksibel dalam menghadapi variasi serangan yang memanipulasi aspek waktu dan distribusi sumber, termasuk serangan *brute force* dengan teknik *slow-rate*.

2.6 *Long Short-Term Memory* (LSTM)

Pada penelitian ini, pendekatan pembelajaran mesin digunakan sebagai model yang diusulkan untuk menganalisis pola autentikasi SSH yang terekam pada log sistem. Model yang dipilih dalam penelitian ini adalah *Long Short-Term Memory* (LSTM), yaitu salah satu arsitektur *Recurrent Neural Network* (RNN) yang dirancang khusus untuk memodelkan data sekuensial dan temporal.

LSTM pertama kali diperkenalkan oleh Hochreiter dan Schmidhuber pada tahun 1997 sebagai solusi atas permasalahan *vanishing gradient* yang umum terjadi pada arsitektur RNN konvensional [28]. Pada RNN standar, informasi dari langkah waktu sebelumnya cenderung hilang ketika sekuens menjadi panjang, sehingga model sulit mempertahankan informasi penting dalam jangka waktu yang lama. LSTM dirancang untuk mengatasi permasalahan ini dengan menambahkan mekanisme memori internal yang dikendalikan oleh beberapa gerbang (*gates*), sehingga informasi relevan dapat dipertahankan atau dilupakan secara selektif.



Gambar 2.5 Struktur Dasar Unit LSTM.
Sumber: [29]

Pada gambar 2.5, LSTM terdiri dari unit memori yang disebut *cell state*, yang berfungsi sebagai jalur utama untuk menyimpan informasi jangka panjang. Aliran informasi dalam *cell state* diatur oleh tiga komponen utama, yaitu *forget gate*, *input gate*, dan *output gate*. Struktur ini memungkinkan LSTM untuk mengontrol informasi apa yang perlu disimpan, diperbarui, atau dikeluarkan pada setiap langkah waktu pembelajaran.

Forget gate bertugas untuk menentukan informasi dari *cell state* sebelumnya yang perlu dipertahankan atau dihapus. Gerbang ini menerima masukan berupa *hidden state* sebelumnya (h_{t-1}) dan input saat ini (x_t), kemudian menghasilkan nilai antara 0 dan 1 menggunakan fungsi aktivasi sigmoid. Nilai mendekati 0 menunjukkan bahwa informasi tersebut perlu dilupakan, sedangkan nilai mendekati 1 menunjukkan bahwa informasi tersebut dipertahankan. Secara matematis, *forget gate* dapat dinyatakan sebagai:

$$f_t=\sigma(W_f\cdot[h_{t-1},x_t]+b_f)$$

dimana W_f dan b_f masing-masing merupakan bobot dan bias, serta σ menyatakan fungsi sigmoid [30].

Setelah proses pelupaan, LSTM menentukan informasi baru yang akan disimpan ke dalam *cell state* melalui *input gate*. *Input gate* terdiri dari dua bagian, yaitu gerbang sigmoid yang menentukan informasi mana yang diperbarui, serta lapisan *tanh* yang menghasilkan kandidat nilai baru untuk ditambahkan ke memori. Persamaan matematis *input gate* adalah sebagai berikut:

$$\mathrm{i_t} = \sigma(\mathrm{W_i} \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{\mathrm{C}}_\mathrm{t} = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

Nilai *cell state* kemudian diperbarui dengan menggabungkan hasil dari *forget gate* dan *input gate*:

$$\mathrm{C_t}=\mathrm{f_t}\cdot\mathrm{C_{t\text{-}1}}+\mathrm{i_t}\cdot\tilde{\mathrm{C}}_\mathrm{t}$$

Melalui mekanisme ini, LSTM mampu mempertahankan informasi penting dalam jangka waktu yang panjang, sekaligus memperbarui memori dengan informasi baru yang relevan.

Komponen terakhir adalah *output gate*, yang menentukan bagian mana dari *cell state* yang akan dikeluarkan sebagai *hidden state* pada langkah waktu pembelajaran. *Hidden state* ini kemudian digunakan sebagai masukan untuk langkah waktu berikutnya atau sebagai keluaran model. *Output gate* dirumuskan seperti berikut:

$$\mathrm{o_t}=\sigma(\mathrm{W_o}\cdot[\mathrm{h_{t\text{-}1}},\mathrm{x_t}]+\mathrm{b_o})$$

$$\mathrm{h_t}=\mathrm{o_t}\cdot\tanh(\mathrm{C_t})$$

Dengan kombinasi ketiga gerbang tersebut, LSTM memiliki kemampuan untuk mempelajari hubungan temporal yang kompleks dalam data sekuensial. Informasi tidak hanya diproses berdasarkan kondisi saat ini, tetapi juga dipengaruhi oleh konteks historis yang tersimpan dalam memori internal jaringan.