# Autoencoder LSTM untuk Deteksi Anomali dalam Log Web Server NGINX

Pandu Rafa Panatagama

Informatika, Telkom University Surabaya, Surabaya, Jawa Timur, Indonesia
pandurafapanatagama@student.telkomuniversity.ac.id

***Abstrak***

*Penelitian ini mengeksplorasi penerapan Autoencoder Long Short-Term Memory (LSTM) untuk deteksi anomali dalam log server web NGINX, dengan fokus pada penggunaan dataset yang terdiri dari log lama yang mengandung anomali dan log baru yang telah dibersihkan. Dataset log baru, yang memiliki tingkat kebersihan 90%, digunakan untuk melatih model, sedangkan log lama berfungsi sebagai data uji untuk mendeteksi anomali. Autoencoder LSTM dipilih karena kemampuannya dalam menangkap pola temporal dalam data sekuensial, yang sangat relevan untuk analisis log. Model ini dievaluasi menggunakan metrik seperti Mean Squared Error (MSE). Selain itu, identifikasi anomali juga dilakukan, namun secara terpisah dengan menggunakan metode tradisional, di mana pola serangan seperti brute force, SQL injection, XSS, dan beberapa serangan lainnya diidentifikasi berdasarkan karakteristik yang telah ditentukan. Hasil penelitian menunjukkan bahwa model Autoencoder LSTM berhasil mendeteksi anomali, sementara metode identifikasi tradisional memberikan informasi tambahan mengenai jenis serangan yang terdeteksi.*

***Keywords:*** *Deteksi Anomali, Autoencoder, LSTM, Identifikasi Anomali, Log Server NGINX*

## 1. Pendahuluan

Dalam era digital saat ini, keamanan siber menjadi salah satu perhatian utama bagi organisasi dan individu. Dengan meningkatnya ketergantungan pada teknologi informasi, ancaman terhadap sistem informasi juga semakin kompleks dan beragam. Serangan siber dapat menyebabkan kerugian finansial yang signifikan, kerusakan reputasi, dan hilangnya data penting. Menurut Von Solms dan Van Niekerk, keamanan siber adalah proses dinamis yang terus berkembang untuk melindungi data dan sistem dari ancaman yang semakin kompleks [1].

Salah satu pendekatan yang efektif untuk menjaga keamanan sistem adalah melalui deteksi anomali. Deteksi anomali adalah proses mengidentifikasi pola yang tidak biasa dalam data yang dapat menunjukkan adanya aktivitas mencurigakan atau serangan. Dengan mendeteksi anomali, organisasi dapat mengambil tindakan pencegahan sebelum kerusakan yang lebih besar terjadi. Menurut Chandola dkk., deteksi anomali sangat penting dalam berbagai domain aplikasi seperti keamanan siber, pengawasan kesehatan, dan deteksi penipuan [2].

Web server, sebagai komponen penting dalam infrastruktur TI, menghasilkan data log yang berisi informasi tentang aktivitas pengguna, permintaan, dan respons server. Data log ini merupakan sumber informasi yang kaya untuk analisis keamanan. Namun, volume dan kompleksitas data log yang besar membuatnya sulit untuk dianalisis secara manual. Menurut Du dkk., log analisis berperan penting dalam mendeteksi anomali dan dapat memberikan wawasan yang mendalam tentang aktivitas sistem, memungkinkan deteksi dini dari potensi ancaman [3].

Deteksi anomali dalam data log web server menghadapi beberapa tantangan, termasuk variasi dalam pola lalu lintas, adanya noise dalam data, dan ketidakseimbangan antara data normal dan anomali. Oleh karena itu, diperlukan metode yang dapat secara efektif menangkap pola dalam data sekuensial dan mendeteksi penyimpangan dari pola tersebut. Seperti yang bisa dikutip dari Aggarwal, tantangan utama dalam deteksi anomali adalah menangani data yang besar, bising, dan kompleks [4].

Autoencoder Long Short-Term Memory (LSTM) merupakan salah satu metode yang menjanjikan untuk deteksi anomali dalam data sekuensial. Autoencoder adalah jenis jaringan saraf yang dirancang untuk belajar representasi data yang efisien dengan tujuan mengurangi dimensi data. Sementara itu, LSTM adalah arsitektur jaringan saraf yang mampu mengingat informasi dalam jangka waktu yang lebih lama, sehingga sangat cocok untuk menangani data sekuensial seperti data log. Menurut Malhotra dkk., autoencoder LSTM menunjukkan kemampuan yang sangat baik dalam mendeteksi anomali pada data sekuensial karena kemampuannya menangkap pola temporal secara efektif [5].

Kombinasi antara autoencoder dan LSTM memungkinkan model untuk belajar pola temporal dalam data log, sehingga meningkatkan akurasi deteksi anomali. Dengan menggunakan autoencoder LSTM, model dapat mendeteksi anomali dengan lebih efektif, bahkan dalam kondisi data yang tidak seimbang. Penelitian ini bertujuan untuk mengeksplorasi dan mengembangkan model Autoencoder LSTM untuk mendeteksi anomali dalam data log web server NGINX, dengan harapan dapat memberikan kontribusi dalam meningkatkan keamanan siber.

## 2. Metode

### 2.1. Autoencoder LSTM

Menurut Baldi, autoencoder adalah jenis jaringan saraf yang digunakan untuk belajar representasi data yang efisien, dengan tujuan untuk mengurangi dimensi data dan mendeteksi anomali. Autoencoder terdiri dari dua bagian utama, encoder, yang mengubah input menjadi representasi dengan dimensi lebih kecil, dan decoder, yang mengembalikan representasi tersebut ke bentuk aslinya. Hal ini memungkinkan autoencoder untuk belajar pola dalam data dan mendeteksi penyimpangan dari pola tersebut, yang sangat berguna dalam konteks deteksi anomali [6].

Hochreiter dan Schmidhuber, yang memperkenalkan LSTM, menjelaskan bahwa LSTM dirancang untuk mengatasi masalah vanishing gradient yang sering terjadi pada jaringan saraf tradisional. Dengan kemampuan untuk mengingat informasi dalam jangka waktu yang lebih lama, LSTM sangat cocok untuk menangani data sekuensial [7], seperti data log yang digunakan dalam tugas ini. Menurut Malhotra dkk., kombinasi autoencoder dengan LSTM memungkinkan model untuk menangkap pola temporal dalam data, sehingga meningkatkan akurasi deteksi anomali.

Dalam penelitian oleh Sabokrou dkk., autoencoder LSTM digunakan untuk mendeteksi anomali dalam data jaringan. Hasil penelitian menunjukkan bahwa model ini mampu mendeteksi anomali dengan tingkat akurasi yang tinggi, bahkan dalam kondisi data yang tidak seimbang. Penelitian ini mendukung penggunaan autoencoder LSTM dalam konteks deteksi anomali, karena kemampuannya untuk belajar dari data yang kompleks dan berurutan [8].

Chalapathy dan Chawla menyatakan bahwa evaluasi kinerja model autoencoder LSTM dalam mendeteksi anomali harus dilakukan dengan metrik yang tepat, seperti Mean Squared Error (MSE) dan Mean Absolute Error (MAE). Mereka menekankan pentingnya menetapkan ambang batas yang tepat untuk mengidentifikasi anomali, yang dapat dilakukan dengan menggunakan persentil dari kesalahan rekonstruksi [9]. Hal ini sejalan dengan pendekatan yang digunakan dalam tugas ini untuk mendeteksi anomali berdasarkan kesalahan rekonstruksi.

### 2.2. Pendekatan Deteksi Anomali

Dataset yang digunakan diambil dari dua data log web server NGINX. Data log pertama adalah data log lama (old.log) yang perkiraan berisi sekitar 1 juta baris data. Sedangkan dataset kedua adalah data log baru (new.log) yang merupakan versi bersih dari data log lama, diketahui bahwa tingkat kebersihan data mencapai 90%.

Data log baru (90% bersih) digunakan sebagai data latih. Data yang diambil sebanyak 10000 baris data, dari baris 1 sampai 10000. Sedangkan data log lama (mengandung anomali) digunakan sebagai data uji, dengan jumlah yang sama yaitu 10000 baris data. Data log lama yang diambil untuk pengujian dimulai dari baris ke 100001 untuk menghindari kesamaan pada beberapa baris antara data latih dan uji.

**Table 1.** Fitur data log NGINX yang dibaca

| Fitur | Contoh |
|---|---|
| ip | 10.252.252.30 |
| time | 10/Nov/2023:11:12:31 +0700 |
| request | GET /wp-admin/ HTTP/2.0 |
| status | 200 |
| size | 19698 |
| referrer | https://surabaya.telkomuniversity.ac.id/wp-login.php |
| user_agent | Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appl... |

Data untuk pelatihan dan pengujian model dibaca dari masing-masing data log menggunakan regex untuk mengekstrak informasi penting seperti *ip, time, request, status, size, referrer,* dan *user_agent.* Data ini kemudian disimpan dalam format *DataFrame* untuk memudahkan manipulasi.

Data yang dibaca diproses dengan mengonversi kolom *size* menjadi numerik dan kolom *time* menjadi format datetime. Fitur tambahan seperti jam, hari, dan bulan diekstrak dari kolom waktu untuk memberikan konteks temporal yang lebih baik.

Kolom status, metode, dan URI di-*encode* menggunakan *one-hot encoding*. Proses ini mengubah data kategorikal menjadi format numerik yang dapat diproses oleh model, sehingga model dapat belajar dari fitur-fitur ini. Jadi fitur yang digunakan untuk pelatihan dan pengujian model adalah *size, hour, day, month, status, method,* dan *uri*.

Data dinormalisasi menggunakan *MinMaxScaler* untuk memastikan bahwa semua fitur berada dalam rentang yang sama (0 hingga 1). Normalisasi ini penting untuk meningkatkan kinerja model, terutama dalam konteks jaringan saraf.

Data yang telah dinormalisasi dibentuk menjadi urutan dengan panjang tertentu (*time step*) untuk digunakan dalam model LSTM, dalam model ini digunakan *time step* sebesar 10. Fungsi *create_sequences* digunakan untuk membuat urutan dari data, yang memungkinkan model untuk belajar dari konteks temporal.

Model autoencoder LSTM dibangun dengan beberapa lapisan LSTM. Arsitektur model ini terdiri beberapa lapisan.

- **Encoder**

  Dua lapisan LSTM yang mengubah input menjadi representasi yang lebih kecil. Lapisan pertama adalah LSTM dengan 64 unit, yang berfungsi sebagai *encoder* pertama. Lapisan ini mengubah input sekuensial menjadi representasi yang lebih kecil. Lapisan kedua juga merupakan LSTM, tetapi dengan 32 unit. Fungsi dari lapisan ini adalah untuk mengurangi dimensi representasi lebih lanjut dengan mengeluarkan *output* terakhir dari urutan.

  Setelah itu, lapisan *RepeatVector* ditambahkan. Lapisan ini berfungsi untuk mengulangi *output* dari lapisan sebelumnya, yang merupakan representasi terkompresi, untuk setiap langkah waktu dalam urutan *output*. Ini diperlukan untuk mempersiapkan data untuk dekoder.

- **Decoder**

  Dua lapisan LSTM yang mengembalikan representasi tersebut ke bentuk aslinya. Lapisan ketiga adalah LSTM dengan 32 unit, yang mulai mendekode representasi terkompresi kembali ke bentuk aslinya. Lapisan keempat adalah LSTM dengan 64 unit, yang melanjutkan proses dekoding dengan menambah dimensi *output* kembali ke ukuran yang lebih besar.

- **Output Layer**

  Lapisan ini menerapkan lapisan *Dense* untuk menghasilkan *output* akhir dari model. Dengan menggunakan *TimeDistributed*, lapisan *Dense* diterapkan pada setiap langkah waktu secara terpisah, sehingga *output* model memiliki bentuk yang sama dengan *input*.

Setelah model dibangun, langkah selanjutnya adalah mengompilasi model sebelum dilatih. Model dikompilasi dengan menggunakan optimizer 'adam', yang merupakan algoritma optimasi yang efisien untuk memperbarui bobot model selama pelatihan. Fungsi *loss* yang digunakan adalah MSE (*Mean Squared Error*), yang mengukur seberapa baik model dalam merekonstruksi inputnya. MSE dihitung sebagai rata-rata kuadrat dari perbedaan antara nilai yang diprediksi dan nilai aktual.

Model dilatih menggunakan data pelatihan dengan tujuan untuk meminimalkan kesalahan rekonstruksi antara input dan output. Model dilatih selama 50 epoch, yang berarti model akan melihat seluruh dataset sebanyak 50 kali. Ukuran batch ditetapkan sebesar 32, yang berarti model akan memperbarui bobotnya setelah memproses 32 sampel data. Selain itu, 20% dari data pelatihan digunakan untuk validasi, yang membantu dalam memantau kinerja model selama pelatihan.

Setelah model dilatih, model diuji dengan data yang belum pernah dilihat sebelumnya, yaitu 10000 baris data dari old.log (dataset yang masih mengandung anomali) pada urutan data ke 100000 (dari indeks 100001 sampai 101000) untuk menghasilkan prediksi rekonstruksi.

Kesalahan rekonstruksi dihitung dengan mengambil rata-rata kesalahan absolut antara data asli dan data yang diprediksi. Kesalahan ini digunakan untuk menentukan apakah suatu data dianggap sebagai anomali.

Ambang batas untuk mendeteksi anomali ditentukan berdasarkan persentil 95 dari kesalahan rekonstruksi. Data yang memiliki kesalahan rekonstruksi lebih besar dari ambang batas ini dianggap sebagai anomali.

### 2.3. Pendekatan Identifikasi Anomali

Data yang terdeteksi sebagai anomali dicatat, dan informasi lebih lanjut tentang anomali tersebut diidentifikasi menggunakan pola serangan yang telah ditentukan. Ini termasuk identifikasi serangan *Brute Forc*e, SQL *injection*, XSS, DoS, dan beberapa macam parameter URI berbahaya.

- ***Brute Force***

  Serangan brute force dapat diidentifikasi dengan memantau jumlah upaya login yang gagal dalam periode waktu tertentu, yang menunjukkan adanya upaya untuk mendapatkan akses tidak sah.

  **Metode Identifikasi:**

  1. Memeriksa apakah status HTTP adalah **401**, yang menunjukkan kegagalan otentikasi.
  2. Jika ditemukan bahwa status adalah 401, maka jumlah upaya gagal untuk IP tertentu ditambahkan.
  3. Jika jumlah upaya gagal melebihi ambang batas yang ditentukan (5x), maka anomali dianggap sebagai serangan *brute force*.

- **SQL *Injection***

  SQL injection dapat ditemukan dengan menganalisis pola permintaan yang mencurigakan, yang sering kali mengandung perintah SQL yang tidak biasa.

  **Metode Identifikasi:**

  1. Menggunakan regex untuk mencari pola yang umum digunakan dalam serangan SQL *injection*, seperti kata kunci SQL (UNION, SELECT, INSERT, dll.).

2. Jika pola tersebut ditemukan dalam *request*, maka anomali dianggap sebagai serangan SQL injection.

- **XSS (*Cross-Site Scripting*)**
  XSS dapat diidentifikasi dengan memeriksa input pengguna yang tidak terfilter, yang sering kali mengandung skrip berbahaya.
  **Metode Identifikasi:**
  1. Memeriksa apakah permintaan mengandung tag '<script>' atau encoding URL untuk tag tersebut.
  2. Jika pola ditemukan, maka anomali dianggap sebagai serangan XSS.

- **DoS (*Denial of Service*)**
  Serangan DoS dapat dikenali dengan memantau frekuensi permintaan dari sumber yang sama dalam waktu singkat, yang menunjukkan upaya untuk membanjiri server.
  **Metode Identifikasi:**
  1. Menyimpan waktu permintaan untuk setiap IP dalam *dictionary* '*request_times'*.
  2. Jika IP sudah ada dalam dictionary, waktu permintaan baru ditambahkan.
  3. Hanya permintaan dalam rentang waktu tertentu (60 detik) yang dipertahankan.
  4. Jika jumlah *request* dari IP tersebut melebihi ambang batas yang ditentukan atau *threshold_dos* (60 *request*), maka anomali dianggap sebagai serangan DoS.

- ***Directory Traversal***
  *Directory traversal* dapat diidentifikasi dengan memeriksa permintaan yang mencoba mengakses file di luar direktori yang diizinkan.
  **Metode Identifikasi:**
  1. Memeriksa apakah permintaan mengandung pola yang menunjukkan upaya untuk menavigasi ke direktori atas.
  2. Jika pola ditemukan, maka anomali dianggap sebagai serangan *directory traversal*.

- **Parameter URI Berbahaya**
  Analisis pola URI dapat membantu dalam mendeteksi permintaan yang mencurigakan dan potensi serangan yang lebih kompleks.
  **Metode Identifikasi:**
  1. Mendefinisikan pola-pola yang dianggap mencurigakan dalam *dictionary* '*suspicious_patterns'*.
  2. Untuk setiap pola, diperiksa apakah URI dalam *request* mengandung pola tersebut.
  3. Jika pola ditemukan, maka pesan yang sesuai akan dikembalikan, menandakan adanya potensi serangan.

Pengidentifikasian anomali yang sudah terdeteksi dalam log ini terpisah atau bukan bagian dari dari model autoencoder LSTM. Identifikasi anomali menggunakan cara tradisional yaitu mengiterasi semua anomali yang tercatat dan mengidentifikasinya satu-persatu menggunakan fungsi khusus untuk menidentifikasi serangan-serangan siber tertentu, dengan metode-metode identifikasi serangan yang sudah disiapkan peneliti.

## 3. Pembahasan Hasil

### 3.1. Hasil Pelatihan Model

Model autoencoder LSTM dilatih menggunakan data log yang telah diproses. Selama pelatihan, loss function (*Mean Squared Error*) dipantau untuk menilai kinerja model.

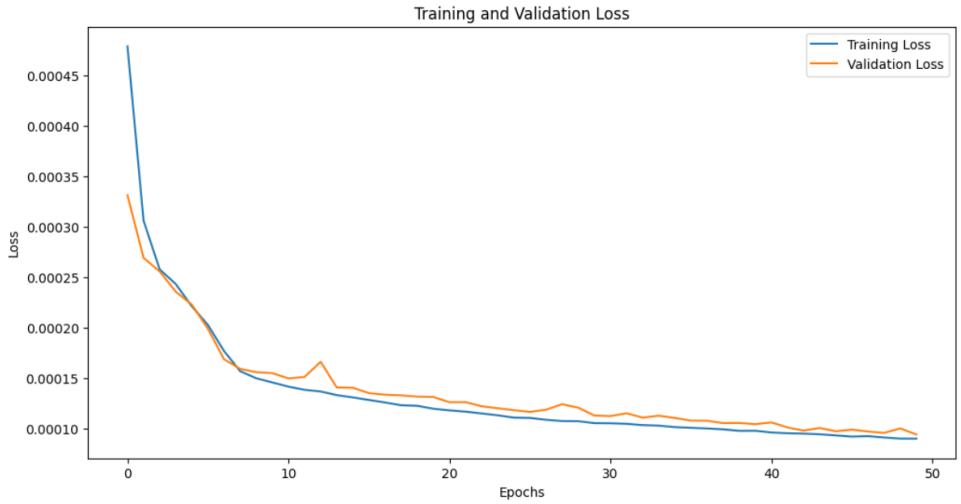


*Figure 1. Kurva loss pelatihan dan validasi dari model Autoencoder LSTM*

```
Epoch 10/50
282/282 ━━━━━━━━━━━━━━━━━━━━ 4s 15ms/step - loss: 1.4649e-04 - val_loss: 1.5493e-04
Epoch 11/50
282/282 ━━━━━━━━━━━━━━━━━━━━ 4s 15ms/step - loss: 1.4319e-04 - val_loss: 1.4966e-04
Epoch 12/50
282/282 ━━━━━━━━━━━━━━━━━━━━ 4s 15ms/step - loss: 1.3977e-04 - val_loss: 1.5116e-04
Epoch 13/50
282/282 ━━━━━━━━━━━━━━━━━━━━ 4s 16ms/step - loss: 1.4044e-04 - val_loss: 1.6616e-04
Epoch 14/50
282/282 ━━━━━━━━━━━━━━━━━━━━ 4s 16ms/step - loss: 1.3585e-04 - val_loss: 1.4071e-04
Epoch 15/50
282/282 ━━━━━━━━━━━━━━━━━━━━ 4s 16ms/step - loss: 1.3181e-04 - val_loss: 1.4041e-04
Epoch 16/50
282/282 ━━━━━━━━━━━━━━━━━━━━ 4s 16ms/step - loss: 1.2861e-04 - val_loss: 1.3509e-04
```

*Image 1. Fluktuasi pada loss validasi di Epoch ke-13*

Model dilatih selama 50 epoch. Model mengalami penurunan loss yang signifikan pada awal pelatihan, baik pada data pelatihan maupun data validasi. Seiring dengan peningkatan epoch, loss pada data pelatihan dan validasi terus menurun, namun dengan laju yang lebih lambat. Terdapat sedikit overfitting pada epoch ke-13, karena loss pada data validasi sedikit meningkat. Namun, hal ini teratasi pada epoch selanjutnya dan loss kembali menurun. Secara keseluruhan, grafik menunjukkan bahwa model berhasil dilatih dengan baik, ditandai dengan penurunan loss yang signifikan pada kedua kurva.

### 3.2. Penentuan *Threshold* (Ambang Batas) Anomali

Ambang batas untuk mendeteksi anomali ditentukan berdasarkan persentil 95 dari kesalahan rekonstruksi. Ini berarti bahwa sampel dengan kesalahan rekonstruksi yang lebih besar dari nilai ambang batas ini dianggap sebagai anomali.

### 3.3. Hasil Pengujian Model

Setelah model dilatih, model diuji menggunakan data log lama untuk mendeteksi anomali. Hasilnya, model menghasilkan prediksi rekonstruksi untuk setiap input dalam data pengujian. *Error* atau kesalahan rekonstruksi dihitung sebagai perbedaan antara input asli dan output yang diprediksi oleh model. Selanjutnya, rata-rata kesalahan rekonstruksi dihitung untuk setiap sampel dalam data pengujian. Kesalahan ini digunakan untuk menentukan apakah suatu sampel dianggap sebagai anomaly berdasarkan ambang batas yang sudah ditentukan.

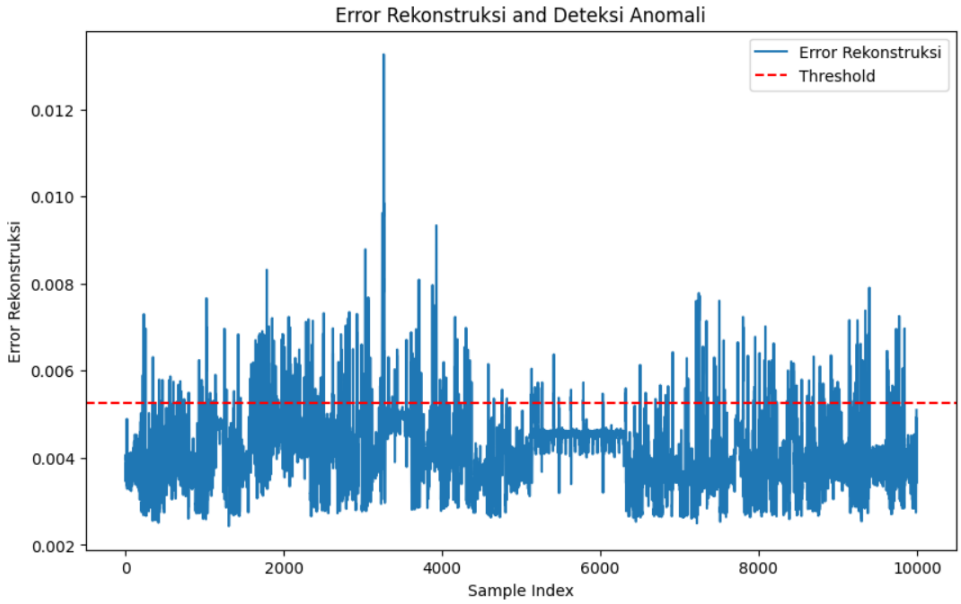


*Figure 2. Grafik Error Rekonstruksi dengan Threshold (Ambang Batas) Anomali (Persentil 95)*

Grafik di atas menunjukkan bahwa terdapat beberapa anomali pada data yang diuji, karena terdapat beberapa titik data yang memiliki error rekonstruksi yang melebihi ambang batas. Ini dapat menunjukkan bahwa data tersebut memiliki karakteristik atau pola yang berbeda dari data bersih yang digunakan dalam pelatihan model (new.log).

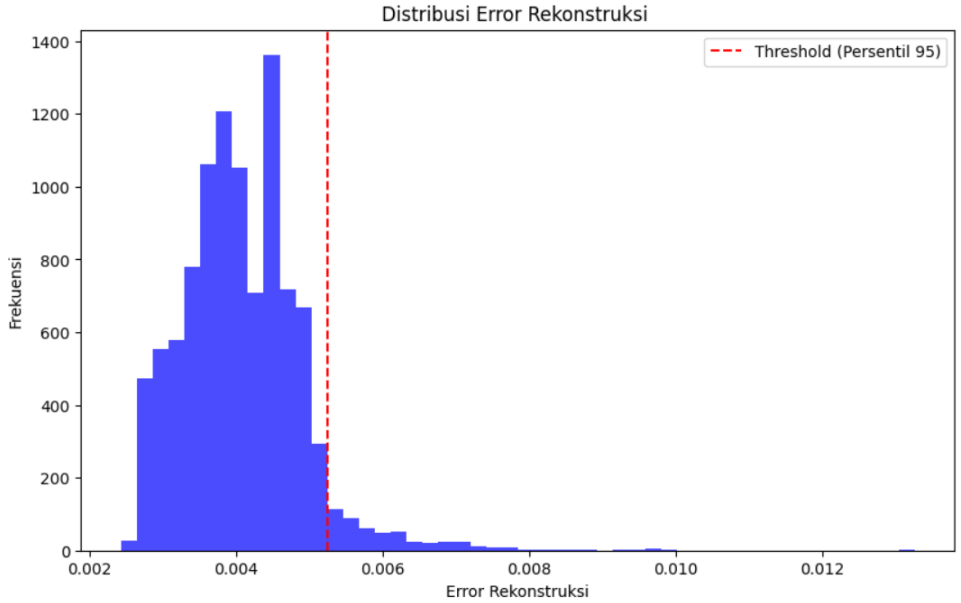


*Figure 3. Histogram Distribusi Error Rekonstruksi*

Histogram di atas lebih memperjelas bahwa sebagian besar error memiliki nilai rendah (sekitar 0.004). Histogram tersebut juga menunjukkan bahwa sebagian besar data dapat direkonstruksi dengan akurasi yang tinggi, tetapi ada beberapa data yang memiliki error yang lebih tinggi, kemungkinan karena anomali atau kompleksitas data tersebut.

### 3.4. Evaluasi Hasil

Setelah model diuji, kinerja deteksi anomali dievaluasi menggunakan beberapa metrik yang umum digunakan dalam analisis regresi. MSE dan RMSE memberikan fokus pada kesalahan yang lebih besar, yang penting dalam deteksi anomali karena kesalahan besar sering kali menunjukkan penyimpangan yang signifikan. Selain itu, MAE menawarkan interpretasi yang lebih stabil dan intuitif tentang kesalahan rata-rata, memberikan ukuran komprehensif dari kinerja model. [9]

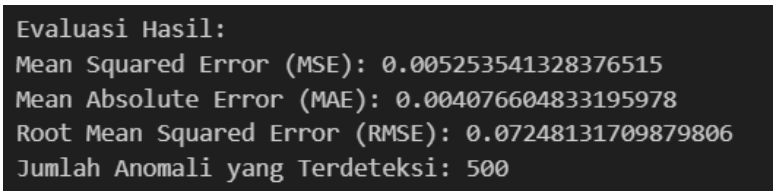


*Image 2. Screenshot Evaluasi Hasil Pengujian Model Autoencoder LSTM*

MSE mengukur rata-rata kuadrat dari kesalahan antara nilai yang diprediksi oleh model dan nilai aktual. Nilai MSE yang rendah, yaitu 0.005, menunjukkan bahwa model mampu merekonstruksi data dengan baik, sehingga kesalahan rekonstruksi relatif kecil.

MAE mengukur rata-rata dari kesalahan absolut antara nilai yang diprediksi dan nilai aktual. Nilai MAE yang rendah, yaitu 0.004, menunjukkan bahwa model memiliki akurasi yang baik dalam memprediksi nilai asli, dengan kesalahan yang dapat diterima.

RMSE adalah akar kuadrat dari MSE dan memberikan gambaran yang lebih intuitif tentang kesalahan dalam satuan yang sama dengan data asli. Nilai RMSE yang rendah, yaitu 0.072, menunjukkan bahwa model dapat mendeteksi anomali dengan akurasi yang baik.

Model berhasil mendeteksi total 500 anomali dalam data pengujian. Jumlah ini memberikan indikasi bahwa model efektif dalam mengidentifikasi penyimpangan dari pola normal dalam data log.

### 3.5. Hasil Indentifikasi Anomali

Setelah model autoencoder LSTM berhasil mendeteksi anomali dalam data log, langkah selanjutnya adalah mengidentifikasi dan memberi label pada anomali yang terdeteksi. Proses ini bertujuan untuk mengklasifikasikan jenis serangan yang mungkin terjadi berdasarkan pola yang telah ditentukan.

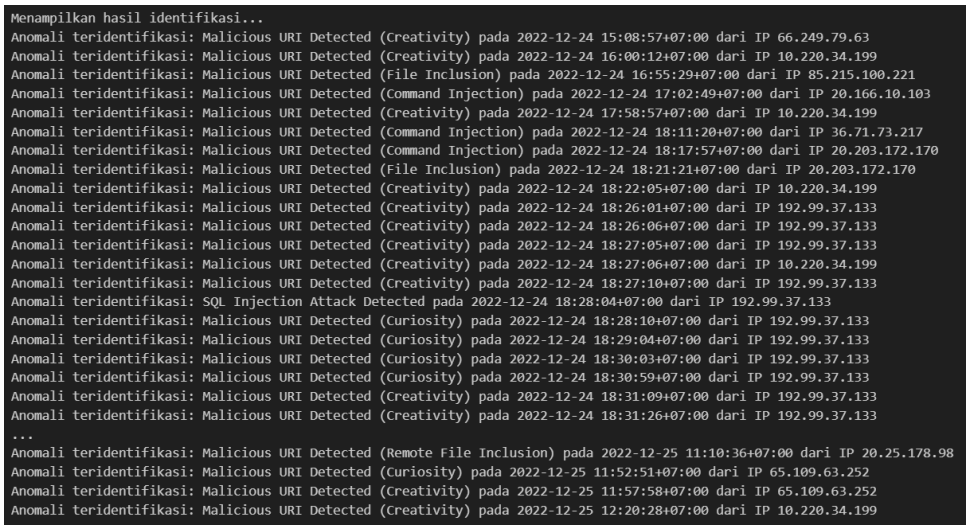


*Image 3. Screenshot Beberapa Hasil Identifikasi Anomali Berdasarkan Label yang ditentukan*

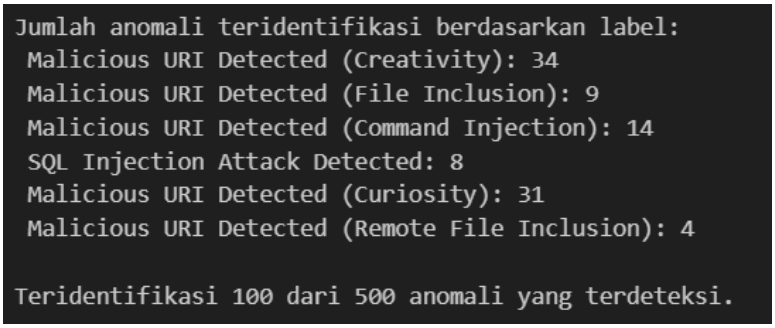


*Image 4. Screenshot Rangkuman Hasil Identifikasi Anomali Berdasarkan Label yang ditentukan*

Dari total 500 anomali yang terdeteksi, 100 anomali berhasil diidentifikasi dan diberi label. Ini berarti bahwa sekitar 20% dari anomali yang terdeteksi dapat diklasifikasikan ke dalam kategori serangan yang spesifik. Persentase ini menunjukkan bahwa meskipun model efektif dalam mendeteksi anomali, ada kemungkinan bahwa beberapa anomali tidak dapat diidentifikasi dengan jelas atau tidak sesuai dengan pola serangan yang telah ditentukan peneliti.

**Distribusi Jenis Serangan:**

- Malicious URI Detected (Creativity) mendominasi dengan 34 kasus, menunjukkan bahwa banyak *request* mencurigakan yang terkait dengan pengaksesan website illegal seperti judi online dan video game bajakan.
- Malicious URI Detected (Curiosity) juga menunjukkan angka yang signifikan dengan 31 kasus, yang menunjukkan adanya permintaan yang mencurigakan berdasarkan rasa ingin tahu pengguna.
- Serangan Command Injection dan File Inclusion masing-masing terdeteksi sebanyak 14 dan 9 kali, menunjukkan bahwa meskipun tidak sebanyak serangan URI, jenis serangan ini tetap menjadi perhatian.
- SQL Injection terdeteksi sebanyak 8 kali, yang menunjukkan bahwa meskipun tidak terlalu umum, serangan ini tetap ada dalam data log.
- Remote File Inclusion terdeteksi paling sedikit dengan 4 kasus, menunjukkan bahwa jenis serangan ini mungkin kurang umum dalam dataset yang dianalisis.

Hasil identifikasi ini memberikan wawasan penting tentang jenis serangan yang mungkin terjadi dalam sistem. Dengan mengetahui jenis serangan yang terdeteksi, tim keamanan dapat mengambil langkah-langkah proaktif untuk memperkuat pertahanan dan mengurangi risiko serangan di masa depan. Identifikasi yang tepat juga memungkinkan untuk analisis lebih lanjut terhadap pola serangan, yang dapat membantu dalam pengembangan strategi mitigasi yang lebih efektif.

## 4. Kesimpulan

Meskipun model yang dihasilkan pada penelitian ini dapat mendeteksi anomali dengan baik, terdapat beberapa keterbatasan yang perlu diperhatikan, seperti data normal yang digunakan untuk pelatihan tidak sepenuhnya normal, yaitu hanya berkisar 90% normal atau bersih. Selain itu kombinasi fitur yang dipilih untuk model mungkin tidak optimal karena dipilih berdasarkan logika peneliti bukan dengan teknik seleksi fitur konvensional. Penelitian selanjutnya disarankan untuk menggunakan data latih yang 100% sudah berlabel normal dan mengeksplorasi teknik seleksi fitur yang mungkin dapat meningkatkan optimasi model, seperti filter methods dan wrapper methods, untuk mengatasi keterbatasan ini. Pengujian model pada dataset yang lebih besar dan beragam juga dapat memberikan wawasan lebih lanjut mengenai efektivitas model dalam mendeteksi anomali. Selain itu, untuk meningkatkan optimasi identifikasi jenis serangan pada anomali yang sudah dideteksi, disarankan untuk menggunakan model klasifikasi tersendiri, seperti Decision Trees, Random Forest, Support Vector Machines (SVM), atau Neural Networks, yang sudah dilatih dengan data anomali pada log NGINX yang memiliki label berbagai serangan siber.

**Daftar Pustaka**


[1] R. Von Solms dan J. Van Niekerk, “From information security to cyber security,” *Comput Secur*, vol. 38, hlm. 97–102, 2013, doi: 10.1016/j.cose.2013.04.004.

[2] V. Chandola, A. Banerjee, dan V. Kumar, “Anomaly Detection: A Survey,” *ACM Comput Surv*, vol. 41, no. 3, hlm. 1–58, Jun 2009, doi: 10.1145/1541880.1541882.

[3] M. Du, F. Li, G. Zheng, dan V. Srikumar, “DeepLog: Anomaly detection and diagnosis from system logs through deep learning,” dalam *Proceedings of the ACM Conference on Computer and Communications Security*, Association for Computing Machinery, Okt 2017, hlm. 1285–1298. doi: 10.1145/3133956.3134015.

[4] C. C. Aggarwal, *Outlier Analysis*, 2 ed. Springer Cham, 2016. doi: 10.1007/978-3-319-47578-3.

[5] P. Malhotra, L. Vig, G. Shroff, dan P. Agarwal, “Long Short Term Memory Networks for Anomaly Detection in Time Series,” dalam *Proceedings of the 23rd European Symposium on Artificial Neural Networks*, Bruges, Belgium: ESANN, Apr 2015. [Daring]. Tersedia pada: http://www.i6doc.com/en/.

[6] P. Baldi, “Autoencoders, Unsupervised Learning, and Deep Architectures,” dalam *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning workshop*, Washington, USA: JMLR.org, Jul 2011, hlm. 37–50. doi: 10.5555/3045796.3045801.

[7] S. Hochreiter dan J. ¨ Urgen Schmidhuber, “Long Short-Term Memory,” *Neural Comput*, vol. 9, no. 8, hlm. 1735–1780, Nov 1997, doi: 10.1162/neco.1997.9.8.1735.

[8] M. Sabokrou, M. Khalooei, M. Fathy, dan E. Adeli, “Adversarially Learned One-Class Classifier for Novelty Detection,” dalam *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, Des 2018, hlm. 3379–3388. doi: 10.1109/CVPR.2018.00356.

[9] R. Chalapathy dan S. Chawla, “Deep Learning for Anomaly Detection: A Survey,” Jan 2019, doi: 10.48550/arXiv.1901.03407.