SESTEM LOG WEB SERVER SEBAGAI PENDETEKSI ANOMALI MENGGUNAKAN ISOLATION FOREST

WEB SERVER LOG SYSTEM AS AN ANOMALY DETECTOR USING ISOLATION FOREST

Dwi Budi Santoso[1]*, Yuli Wahyuni[2]
[1] Progam Studi Sistem Informasi, Fakultas Teknologi Informasi dan Industri, Universitas Stikubank
[2]Progam Studi Teknik Komputer, Fakultas Sekolah Vokasi, Universitas Pakuan
[1]dbs@edu.unisbank.ac.id, [2]yuli_wahyuni@unpak.ac.id

ABSTRAK

Deteksi anomali pada *log* sistem *web server* merupakan langkah penting dalam memastikan keamanan dan kinerja optimal dari sistem tersebut dan terdapat keterbatasan data yang tidak lengkap, variabilitas pola lalu lintas, kualitas data, overfitting model dan kecepatan pemprosesan. Penelitian ini menggunakan model *Isolation Forest* untuk mendeteksi anomali pada l*og* sistem *web server Apache* yang dikumpulkan selama dua bulan, dengan total 10.041 entri *log* yang dianalisis. Model ini dirancang untuk mengidentifikasi entri yang tidak biasa berdasarkan fitur size dari respons *HTTP*. Hasil penelitian menunjukkan bahwa model *Isolation Forest* berhasil mengidentifikasi 499 entri sebagai anomali, yang mencakup sekitar 5% dari total dataset. Evaluasi kinerja model menggunakan Stratified K-Fold Cross-Validation menghasilkan mean score sebesar -0.3802 dengan standard deviation sebesar 0.0023, yang menunjukkan konsistensi dan stabilitas deteksi anomali di berbagai bagian dataset. Anomali yang terdeteksi cenderung berkaitan dengan ukuran respons yang sangat besar atau status *HTTP* yang menunjukkan kesalahan, seperti 404 (*Not Found*). Berdasarkan pola anomali yang terdeteksi, penelitian ini merekomendasikan untuk melakukan audit mendalam terhadap konfigurasi *server*, terutama untuk mengidentifikasi dan memperbaiki kesalahan yang mungkin menyebabkan ukuran respons yang tidak biasa. Selain itu, disarankan untuk memonitor aktivitas dari IP yang mencurigakan secara lebih ketat, guna mencegah potensi serangan yang dapat merusak integritas dan kinerja *server*. Penelitian ini menunjukkan bahwa *Isolation Forest* adalah alat yang efektif untuk deteksi anomali dalam l*og* sistem *web server* dan dapat digunakan sebagai bagian dari strategi pemantauan keamanan yang lebih luas.

Kata kunci : Deteksi Anomali, *Log* Sistem, *Isolation Forest*, Keamanan *Server*

ABSTRACT

Anomaly detection in web server system logs is an important step in ensuring the security and optimal performance of such systems and there are limitations of incomplete data, traffic pattern variability, data quality, model overfitting and processing speed. This study uses the Isolation Forest model to detect anomalies in Apache web server system logs collected over two months, with a total of 10,041 log entries analyzed. The model is designed to identify unusual entries based on the size features of HTTP responses. The results showed that the Isolation Forest model successfully identified 499 entries as anomalous, which accounted for about 5% of the total dataset. Evaluation of the model's performance using Stratified K-Fold Cross-Validation resulted in a mean score of -0.3802 with a standard deviation of 0.0023, indicating consistency and stability of anomaly detection across different parts of the dataset. The detected anomalies tend to be related to very large response sizes or HTTP statuses that indicate errors, such as 404 (Not Found). Based on the detected anomaly patterns, this study recommends conducting an in-depth audit of the server configuration, especially to identify and fix errors that might cause unusual response sizes. In addition, it is recommended to monitor the activity of suspicious IPs more closely, in order to prevent potential attacks that could damage the integrity and performance of the server. This research shows that Isolation Forest is an effective tool for anomaly detection in web server system logs and can be used as part of a broader security monitoring strategy.

Keywords: Anomaly Detection, System Log , Isolation Forest, Server Security

PENDAHULUAN

Dalam era digital yang semakin berkembang, *web server* menjadi salah satu komponen kritis dalam infrastruktur TI yang mendukung berbagai aplikasi dan layanan online. *Web server* seperti *Apache*, yang merupakan salah satu *web server* paling populer di dunia, digunakan secara luas oleh organisasi besar maupun kecil untuk mengelola lalu lintas web, menangani permintaan *HTTP*, dan menyajikan konten kepada pengguna [1], [11]. Meskipun *Apache* dikenal andal, kompleksitas dan skala operasinya membuatnya rentan terhadap berbagai masalah, seperti serangan siber, kegagalan sistem, dan kesalahan konfigurasi [2]. *Log* sistem adalah sumber data penting yang merekam berbagai aktivitas yang terjadi pada *server*, termasuk permintaan yang diterima, respon yang diberikan, serta berbagai peringatan dan kesalahan yang mungkin terjadi [3]. Analisis terhadap *Log* ini dapat memberikan wawasan mendalam tentang kesehatan dan kinerja *web server*. Namun, dalam lingkungan yang sibuk, di mana ribuan hingga jutaan entri *Log* dihasilkan setiap harinya, mendeteksi aktivitas abnormal atau anomali secara manual menjadi tugas yang hampir mustahil.

Oleh karena itu, diperlukan pendekatan otomatis untuk mendeteksi anomali dalam *log* sistem. Pendekatan ini tidak hanya membantu dalam mendeteksi dan mencegah potensi ancaman seperti serangan *Distributed Denial of Service (DDoS)* atau upaya penetrasi sistem, tetapi juga membantu dalam mengidentifikasi masalah kinerja yang dapat menyebabkan *downtime* atau penurunan kualitas layanan. Penelitian mengenai deteksi anomali dalam *log* sistem telah dilakukan oleh banyak peneliti dengan berbagai pendekatan. Salah satu pendekatan tradisional yang banyak digunakan adalah berbasis aturan (*rule-based*), di mana sistem akan mendeteksi anomali berdasarkan aturan yang telah ditentukan sebelumnya [4]. Namun, metode ini memiliki kelemahan, terutama dalam menghadapi pola serangan yang baru dan tidak terduga, serta dalam menangani volume data yang besar dan kompleksitas sistem yang tinggi.

Dalam beberapa tahun terakhir, algoritma berbasis ensemble seperti *Isolation Forest* telah mendapatkan perhatian karena kemampuannya dalam mendeteksi *outlier* tanpa memerlukan pelabelan data [5]. Hasil penelitian juga menunjukkan bahwa algoritma ini efektif dalam mendeteksi anomali pada dataset besar dan berdimensi tinggi dan dalam berbagai domain, termasuk keamanan siber dan pemantauan kinerja jaringan, dengan hasil yang menjanjikan [6]. Namun, penerapan *Isolation Forest* untuk deteksi anomali dalam *log* sistem *web server Apache* masih relatif jarang ditemukan dalam literatur. Penelitian ini berupaya untuk mengisi celah tersebut dengan menerapkan *Isolation Forest* untuk mendeteksi anomali pada *log* sistem *Apache*, dengan fokus pada identifikasi pola-pola yang tidak biasa yang dapat mengindikasikan adanya masalah pada *server*, baik itu terkait dengan serangan siber, kesalahan konfigurasi, maupun anomali kinerja lainnya. Penelitian ini bertujuan untuk mengembangkan dan menguji pendekatan berbasis *Isolation Forest* untuk deteksi anomali dalam l*og* sistem *Apache*. Dengan menganalisis *log* ini, akan dilakukan identifikasi pola-pola yang tidak biasa yang dapat mengindikasikan adanya masalah pada *server*, baik dari segi keamanan maupun kinerja. Hasil dari penelitian ini diharapkan dapat memberikan kontribusi signifikan dalam pengembangan teknik pemantauan otomatis yang lebih efektif dan efisien untuk *server Apache*.

METODE PENELITIAN

Penelitian ini dilakukan untuk mendeteksi anomali pada *log* sistem *web server Apache* menggunakan algoritma *Isolation Forest*. Metode penelitian ini terdiri dari beberapa tahap utama, yaitu pengumpulan data, praproses data, penerapan model *Isolation Forest*, evaluasi kinerja model menggunakan *cross-validation*, dan analisis hasil. Setiap tahap dijelaskan secara rinci sebagai berikut:

1. Pengumpulan Data

Data yang digunakan dalam penelitian ini adalah access *log* yang dihasilkan oleh *web server Apache*, yang mencatat semua permintaan *HTTP* yang diterima oleh *server*, termasuk informasi seperti alamat IP pengunjung, waktu permintaan, metode *HTTP* yang digunakan, URL yang diminta, status respon, dan ukuran respons. Access l*og* ini dikumpulkan selama periode 1 Juli 2023 hingga 31 Agustus 2023, mencakup total dua bulan operasional *server Apache*, dengan jumlah total sebanyak 10.041 entri *log* dalam file access. *log* . Beberapa contoh entri dari access l*og* tersebut ditampilkan dalam Gambar 1. untuk menunjukkan struktur dan jenis informasi yang dianalisis.

```
114.119.137.160 - - [03/Sep/2024:00:00:04 +0000] "GET /index.php/b85443d181q8sz38_051ar_d/aeaaccf/07368317 HTTP/1.1"
301 4021 "-" "Mozilla/5.0 (Linux; Android 7.0;) AppleWebKit/537.36 (KHTML, like Gecko) Mobile Safari/537.36
(compatible; PetalBot;+https://webmaster.petalsearch.com/site/petalbot)"
51.222.253.14 - - [03/Sep/2024:00:00:09 +0000] "GET /pulsa/googlef0880f9e814d30f0.html HTTP/1.1" 200 3608 "-"
"Mozilla/5.0 (compatible; AhrefsBot/7.0; +http://ahrefs.com/robot/)"
114.119.135.163 - - [03/Sep/2024:00:00:19 +0000] "GET /index.php/b26529467P56_85PUH586r/aeccacd/54649004 HTTP/1.1" 301
4017 "-" "Mozilla/5.0 (Linux; Android 7.0;) AppleWebKit/537.36 (KHTML, like Gecko) Mobile Safari/537.36 (compatible;
PetalBot;+https://webmaster.petalsearch.com/site/petalbot)"
114.119.137.160 - - [03/Sep/2024:00:00:34 +0000] "GET /?beaace012I69_4Ij81947r_d/dedead979126507 HTTP/1.1" 301 4001
"-" "Mozilla/5.0 (Linux; Android 7.0;) AppleWebKit/537.36 (KHTML, like Gecko) Mobile Safari/537.36 (compatible;
PetalBot;+https://webmaster.petalsearch.com/site/petalbot)"
168.235.86.13 - - [03/Sep/2024:00:00:39 +0000] "GET / HTTP/1.1" 302 3629 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X
10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36"
185.224.128.59 - - [03/Sep/2024:00:00:41 +0000] "GET
/cgi-bin/luci/;stok=/locale?form=country&operation=write&country=$(id%3E%60wget+-O-+http%3A%2F%2F154.216.17.171%3A88%2F
t%7Csh%3B%60) HTTP/1.1" 404 3750
"http://103.252.188.21:80/cgi-bin/luci/;stok=/locale?form=country&operation=write&country=$(id%3E%60wget+-O-+http%3A%2F
%2F154.216.17.171%3A88%2Ft%7Csh%3B%60)" "Go-http-client/1.1"
52.167.144.229 - - [03/Sep/2024:00:00:50 +0000] "GET /beefdc2 HTTP/1.1" 301 4020 "-" "Mozilla/5.0 AppleWebKit/537.36
(KHTML, like Gecko; compatible; bingbot/2.0; +http://www.bing.com/bingbot.htm) Chrome/116.0.1938.76 Safari/537.36"
114.119.155.210 - - [03/Sep/2024:00:00:50 +0000] "GET /?be0577783sf8m834-538cr/dedaefa90164059 HTTP/1.1" 301 3997 "-"
"Mozilla/5.0 (Linux; Android 7.0;) AppleWebKit/537.36 (KHTML, like Gecko) Mobile Safari/537.36 (compatible;
PetalBot;+https://webmaster.petalsearch.com/site/petalbot)"
114.119.140.84 - - [03/Sep/2024:00:01:04 +0000] "GET /?bd921a3d3_53v79ll1u5d8r/dcdafcad/8331950 HTTP/1.1" 301 4001 "-"
"Mozilla/5.0 (Linux; Android 7.0;) AppleWebKit/537.36 (KHTML, like Gecko) Mobile Safari/537.36 (compatible;
PetalBot;+https://webmaster.petalsearch.com/site/petalbot)"
92.51.2.78 - - [03/Sep/2024:00:01:11 +0000] "GET /wp-includes/js/jquery/jquery-migrate.min.js?ver=3.4.1 HTTP/1.1" 200
8282 "-" "Mozilla/5.0 (Windows; U; Windows NT 6.0; fr-FR) AppleWebKit/533.18.1 (KHTML, like Gecko) Version/5.0.2
Safari/533.18.5"
```

Gambar 1. Sampel Entri *Access log*

2. Praproses Data

Sebelum data access *log* dapat digunakan dalam model machine learning, beberapa langkah praproses dilakukan untuk memastikan data berada dalam format yang sesuai dan siap dianalisis[7]. Langkah pertama adalah parsing *log* , di mana data *log* diuraikan menjadi komponen-komponen utama seperti *Remote_IP, Timestamp, Request, Status, Size, Referer,* dan *User_Agent*. Langkah ini penting untuk mengekstraksi informasi inti dari setiap entri *log* yang diperlukan dalam analisis lebih lanjut.

Setelah itu, dilakukan konversi waktu pada kolom Timestamp agar dapat diubah menjadi format datetime yang lebih mudah digunakan dalam analisis lanjutan. Selain itu, kolom Size yang menunjukkan ukuran respons dikonversi menjadi tipe data numerik, dan nilai "-" yang menunjukkan data tidak tersedia diganti dengan 0. Ini dilakukan untuk memastikan semua data memiliki format yang konsisten dan dapat diolah oleh model.

Langkah terakhir dalam praproses data adalah ekstraksi metode, *URL*, dan versi *HTTP* dari kolom *Request*, yang dipecah menjadi tiga bagian: *Method, URL, dan HTTP Version*. Dari semua komponen yang dicatat dalam *access. log* hanya 5 yang dipakai dalam penelitian ini, yaitu *Size, Timestamp, Status, Method* dan *URL*. Sampel data yang sudah di praproses seperti ditampilkan pada tabel 1.

Tabel 1. Sampel Data Hasil Praproses

Timestamp	Status	Size	Method	URL
2024-09-03 02:48:51+00:00	304	3255	GET	/wp-content/uploads/2021/06/WhatsApp-Image-202...
2024-09-03 02:52:46+00:00	304	3268	GET	/wp-content/uploads/2020/10/1-j2GtIrbQBiYiAwBG...
2024-09-03 05:32:56+00:00	304	3253	GET	/wp-content/themes/personal-portfolio/inc/js/s...
2024-09-03 01:48:46+00:00	301	4025	GET	/index.php?b42c1de8195-N8t70i533r_d/dfceccdfa/...
2024-09-03 02:24:16+00:00	404	3857	GET	/wp-admin/css/colors/coffee/-/superkoin88/NEW/...

3. Penerapan Model *Isolation Forest*

Setelah data dipraproses, algoritma *Isolation Forest* diterapkan untuk mendeteksi anomali dalam data *log*. *Isolation Forest* adalah algoritma berbasis ensemble yang dirancang khusus untuk mendeteksi *outlier* atau anomali dalam dataset[8], [9], [10]. Berbeda dengan model lain yang menggunakan pendekatan berbasis jarak atau kepadatan, *Isolation Forest* bekerja dengan cara mengisolasi setiap sampel data secara acak. Konsep utamanya adalah bahwa anomali adalah poin data yang lebih mudah diisolasi dibandingkan dengan data normal, karena anomali cenderung berada jauh dari kumpulan data utama.

Cara kerja *Isolation Forest* dimulai dengan membuat beberapa pohon keputusan secara acak. Setiap pohon dibangun dengan memilih fitur secara acak dan kemudian membagi data berdasarkan nilai acak dari fitur tersebut. Proses ini berulang hingga setiap sampel data benar-benar terisolasi dalam sebuah node. Kedalaman rata-rata di mana suatu sampel terisolasi di semua pohon digunakan untuk menghitung skor anomali. Semakin cepat sebuah sampel terisolasi (yaitu, semakin rendah kedalamannya), semakin besar kemungkinan bahwa sampel tersebut adalah anomali.

Secara matematis, skor anomali untuk sebuah sampel x dapat dihitung sebagai:

$$S(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

dimana :

- *E(h(x))* adalah kedalaman rata-rata dari sampel x di semua pohon dalam hutan.
- *c(n)* adalah fungsi yang mendekati kedalaman rata-rata untuk pohon yang sepenuhnya dibangun dan digunakan untuk normalisasi.

Nilai skor ini berada pada rentang [-1, 1], dengan nilai mendekati -1 menunjukkan kemungkinan besar anomali.

Langkah-langkah dalam penerapan model ini meliputi:

- Pelatihan Model: Model *Isolation Forest* dilatih menggunakan keseluruhan dataset, yang secara otomatis mengidentifikasi data yang dianggap anomali tanpa memerlukan label pelatihan. Karena model ini unsupervised, tidak diperlukan pembagian eksplisit antara data latih dan uji.
- Deteksi Anomali: Setelah pelatihan, model memberikan prediksi untuk setiap sampel data dengan menentukan apakah sampel tersebut adalah anomali (−1) atau normal (1). Prediksi ini didasarkan pada *outlier* scores yang dihasilkan oleh model, di mana sampel yang memiliki skor tinggi dianggap sebagai anomali.

4. Evaluasi Kinerja

Evaluasi kinerja model dilakukan menggunakan metode cross-validation. Dalam proses ini, dataset dibagi menjadi beberapa subset menggunakan Stratified K-Fold Cross-Validation dengan 5 fold. Pada setiap fold, model *Isolation Forest* dilatih pada subset tertentu dan kemudian diuji pada subset yang berbeda. *Outlier* scores dihitung pada setiap subset uji untuk mengukur seberapa baik model mendeteksi anomali di seluruh dataset. Setelah itu, skor rata-rata dari *outlier* scores dihitung untuk setiap fold, memberikan gambaran umum tentang kinerja model secara keseluruhan dalam mendeteksi anomali. Selain itu, standard deviation dari skor di berbagai fold juga dihitung untuk mengukur konsistensi kinerja model di seluruh bagian dataset. Evaluasi ini penting untuk memastikan bahwa model memiliki performa yang stabil dan andal ketika diterapkan pada berbagai bagian data.

5. Analisis Hasil

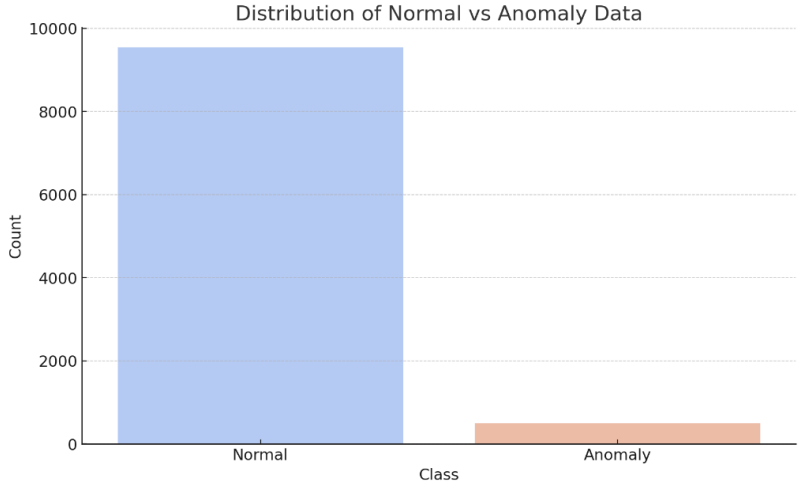
Hasil deteksi anomali dan evaluasi model dianalisis untuk memahami distribusi anomali serta efektivitas model dalam mendeteksi data yang tidak biasa. Pertama, dilakukan identifikasi pola anomali, yaitu meninjau data yang diidentifikasi sebagai anomali oleh model untuk memahami pola-pola mendasar yang menyebabkan deteksi tersebut. Hal ini membantu dalam menemukan karakteristik umum dari entri yang dianggap anomali. Selain itu, distribusi antara data normal dan anomali divisualisasikan untuk melihat proporsi anomali dalam dataset secara keseluruhan, memberikan gambaran jelas tentang seberapa sering anomali muncul dalam data. Terakhir, konsistensi model dievaluasi melalui hasil cross-validation untuk menilai apakah model memiliki performa yang stabil dalam mendeteksi anomali di berbagai bagian dataset.

HASIL DAN PEMBAHASAN

Hasil dari penelitian ini dibagi menjadi beberapa bagian, yaitu distribusi anomali dalam dataset, hasil deteksi anomali, evaluasi kinerja model, dan analisis pola anomali.

1. Distribusi Anomali dalam Dataset

- Dari total 10.041 entri *log* yang dianalisis, model *Isolation Forest* mengidentifikasi 499 entri sebagai anomali dan 9.542 entri sebagai data normal.
- Distribusi ini menunjukkan bahwa sekitar 5% dari total data dianggap sebagai anomali, yang konsisten dengan parameter kontaminasi yang ditetapkan dalam model (contamination=0.05).
- Visualisasi distribusi seperti ditampilkan pada gambar 2, menunjukkan bahwa anomali merupakan sebagian kecil dari dataset, namun penting untuk diperhatikan karena dapat mengindikasikan aktivitas yang tidak biasa atau berbahaya.



Gambar 2. Visualisasi Distribusi Anomali

2. Hasil Deteksi Anomali

Model *Isolation Forest* menggunakan fitur size dari respons *HTTP* untuk mendeteksi anomali dalam data. Entri l*og* yang memiliki ukuran respons sangat besar atau kecil dibandingkan dengan mayoritas data lainnya cenderung diidentifikasi sebagai anomali. Beberapa contoh anomali yang terdeteksi melibatkan permintaan GET dengan ukuran respons yang tidak biasa, seperti lebih dari 283.000 bytes, yang jauh di atas ukuran rata-rata respons *HTTP* normal. Selain itu, entri dengan status *HTTP* 404 (Not Found) dan ukuran respons yang besar juga sering diidentifikasi sebagai anomali, yang mungkin mengindikasikan upaya akses ke halaman yang tidak ada atau adanya kesalahan konfigurasi.

3. Evaluasi Kinerja Model Menggunakan Cross-Validation

Kinerja model dievaluasi menggunakan metode Stratified K-Fold Cross-Validation dengan 5 fold untuk mengukur konsistensi deteksi anomali pada berbagai bagian dataset. Rata-rata skor *outlier* di seluruh fold (Mean Score) adalah -0.3802, yang mengindikasikan bahwa data yang diidentifikasi sebagai anomali memiliki perbedaan signifikan dibandingkan dengan data normal. Standard deviation dari hasil cross-validation sebesar 0.0023 menunjukkan bahwa model memberikan hasil yang sangat konsisten di setiap fold, dengan variasi yang sangat kecil dalam kinerja deteksi anomali. Hasil ini mengindikasikan bahwa model *Isolation Forest* memiliki performa yang stabil dalam mendeteksi anomali di seluruh dataset, tanpa terlalu bergantung pada bagian data tertentu, sehingga memberikan keyakinan akan keandalan model dalam mendeteksi pola yang tidak biasa.

4. Analisis Pola Anomali

Karakteristik anomali yang terdeteksi dalam penelitian ini umumnya berkaitan dengan ukuran respons *HTTP* yang tidak biasa. Ukuran respons yang sangat besar atau sangat kecil dibandingkan dengan nilai rata-rata pada data normal sering kali diidentifikasi sebagai anomali. Selain itu, anomali juga sering ditemukan pada permintaan dengan status *HTTP* yang menunjukkan kesalahan, seperti kode 404 (Not Found), yang mengindikasikan upaya akses ke halaman yang tidak ada atau salah konfigurasi. Kombinasi antara ukuran respons yang tidak wajar dan status *HTTP* yang menunjukkan kesalahan merupakan faktor utama dalam deteksi anomali ini.

Pola yang berulang ditemukan pada entri *log* yang diidentifikasi sebagai anomali. Beberapa alamat IP menunjukkan perilaku yang konsisten dalam mengirimkan permintaan dengan ukuran respons yang sangat besar secara berulang-ulang. Pola ini dapat mengindikasikan adanya upaya serangan atau kesalahan sistem yang terjadi secara berkala. Aktivitas semacam ini penting untuk dianalisis lebih lanjut, karena potensi risiko keamanan atau masalah kinerja sistem mungkin terjadi sebagai hasil dari perilaku anomali yang terus berulang.

Berdasarkan pola anomali yang teridentifikasi, disarankan untuk melakukan audit menyeluruh terhadap konfigurasi *server* guna memastikan tidak adanya kesalahan signifikan yang dapat mempengaruhi kinerja atau keamanan sistem. Selain itu, monitoring yang lebih ketat terhadap alamat IP yang mencurigakan perlu dilakukan untuk mencegah potensi serangan lebih lanjut. Pemantauan secara berkelanjutan terhadap pola-pola anomali ini dapat membantu dalam mengidentifikasi masalah keamanan atau operasional lebih dini, sehingga tindakan pencegahan dapat segera diambil.

KESIMPULAN DAN SARAN

Penelitian ini berhasil menunjukkan bahwa model *Isolation Forest* efektif dalam mendeteksi anomali pada *log* sistem *web server Apache*. Dengan menggunakan fitur ukuran respons dan status *HTTP*, model mampu mengidentifikasi entri *log* yang berbeda secara signifikan dari data normal. Hasil evaluasi melalui Stratified K-Fold Cross-Validation menunjukkan bahwa model memiliki kinerja yang konsisten dan stabil dalam mendeteksi anomali, dengan mean score sebesar -0.3802 dan standard deviation sebesar 0.0023. Pola

anomali yang ditemukan, seperti ukuran respons yang tidak biasa dan status *HTTP* kesalahan, memberikan indikasi bahwa model ini dapat digunakan secara efektif untuk mendeteksi aktivitas yang mencurigakan atau masalah dalam konfigurasi *server*.

Berdasarkan hasil penelitian, disarankan untuk melakukan audit berkala terhadap konfigurasi *server*, khususnya terkait dengan permintaan *HTTP* yang berulang dengan ukuran respons yang tidak wajar atau status kesalahan 404. Selain itu, monitoring yang lebih intensif terhadap alamat IP yang menunjukkan perilaku anomali perlu dilakukan guna mencegah potensi serangan atau gangguan operasional. Implementasi sistem pemantauan otomatis berbasis machine learning dapat meningkatkan deteksi dini terhadap anomali dan membantu menjaga keamanan serta kinerja *server* secara optimal.

DAFTAR PUSTAKA

[1] A. Y. Chandra, "Analisis performansi antara apache & nginx web server dalam menangani client request," *Jurnal Sistem Dan Informatika (JSI)*, vol. 14, no. 1, hlm. 48–56, 2019.

[2] R. Ramadhan, J. Latuny, dan S. J. Litiloly, "Perancangan Pengamanan Server Apache Menggunakan Firewall Iptables Dan Fail2ban," *J. ISOMETRI*, vol. 1, no. 1, hlm. 9–15, 2022.

[3] B. Wijaya dan A. Pratama, "Deteksi Penyusupan Pada Server Menggunakan Metode Intrusion Detection System (Ids) Berbasis Snort," *Jurnal Sisfokom (Sistem Informasi Dan Komputer)*, vol. 9, no. 1, hlm. 97–101, 2020.

[4] L. Decker, D. Leite, L. Giommi, dan D. Bonacorsi, "Real-time anomaly detection in data centers for log-based predictive maintenance using an evolving fuzzy-rule-based approach," dalam *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2020, hlm. 1–8.

[5] J. Lesouple, C. Baudoin, M. Spigai, dan J.-Y. Tourneret, "Generalized isolation forest for anomaly detection," *Pattern Recognit Lett*, vol. 149, hlm. 109–119, 2021.

[6] M. Carletti, M. Terzi, dan G. A. Susto, "Interpretable anomaly detection with diffi: Depth-based feature importance of isolation forest," *Eng Appl Artif Intell*, vol. 119, hlm. 105730, 2023.

[7] D. Ridhwanullah, "Pemodelan Topik pada Cuitan tentang Penyakit Tropis di Indonesia dengan Metode Latent Dirichlet Allocation," 2022.

[8] H. Chen, H. Ma, X. Chu, dan D. Xue, "Anomaly detection and critical attributes identification for products with multiple operating conditions based on isolation forest," *Advanced Engineering Informatics*, vol. 46, hlm. 101139, 2020.

[9] M. T. R. Laskar *dkk.*, "Extending isolation forest for anomaly detection in big data via K-means," *ACM Transactions on Cyber-Physical Systems (TCPS)*, vol. 5, no. 4, hlm. 1–26, 2021.

[10] C. Li, L. Guo, H. Gao, dan Y. Li, "Similarity-measured isolation forest: Anomaly detection method for machine monitoring data," *IEEE Trans Instrum Meas*, vol. 70, hlm. 1–12, 2021.

[11] M. D. Zebua, S. Maryana, Y. Wahyuni. "Sistem Monitoring Network Menggunakan Zabbix Pada Linux Ubuntu." Jurnal Aplikasi Bisnis dan Komputer 4, no. 2 : 76-82. 2024.