Journal of Applied Informatics and Computing (JAIC)
Vol.x, No.x, Juli 201x, pp. xx~xx
e-ISSN: 2548-6861

Perancangan dan Implementasi Sistem Deteksi dan Manajemen Insiden Keamanan Siber Berbasis Log Autentikasi SSO Pada PT. XYZ

Vincentius Simanjuntak [1]*, Antoni Haikal [2]**
* Teknik Informatika, Politeknik Negeri Batam
** Rekayasa Kemanan Siber, Politeknik Negeri Batam
vincentius.4332211002@students.polibatam.ac.id [1], antoni@polibatam.ac.id [2]

Article Info

Article history:

Received …
Revised …
Accepted …

Keywords:

Single Sign-On; Keycloak; brute force; credential stuffing; log analysis; Redis; TheHive; incident management; deduplication

ABSTRACT

The advancement of information technology has driven the widespread adoption of Single Sign-On (SSO) as a centralized authentication solution, making it a prime target for attacks such as brute force and credential stuffing. This study proposes the design and implementation of a cybersecurity incident detection and management system based on Keycloak SSO authentication log analysis. The system integrates four main components: Keycloak as the SSO provider, Redis for *alert* deduplication and temporary storage, TheHive as the incident management platform, and n8n for workflow automation. Threat detection is performed by a Python worker that continuously reads Keycloak LOGIN_ERROR logs and compares them against defined security rules—brute force and credential stuffing detection. A deduplication mechanism using Redis prevents duplicate incident tickets from being created in TheHive. Experimental testing with simultaneous attack injection from two IP addresses across eight simulation scenarios showed that the system achieved 100% detection accuracy for both attack types. The deduplication mechanism successfully ensured only one unique incident case was created per attacker entity per day. The average end-to-end processing time from *alert* detection to case creation in TheHive was 2.937 seconds. These results demonstrate that the proposed architecture effectively automates security monitoring, *alert* deduplication, and incident ticket creation in an efficient and structured manner.

I. PENDAHULUAN

Perkembangan teknologi informasi mendorong perusahaan untuk melakukan digitalisasi hampir pada setiap proses bisnis dan hal ini mengakibatkan meningkatnya kebutuhan sistem yang aman dan efisien. *Single Sign-On* (SSO) menjadi solusi yang banyak digunakan karena memungkinkan pengguna mengakses beberapa layanan hanya dalam satu kali proses login [1][2].

Meskipun memberikan kemudahan, implementasi SSO juga memiliki risiko keamanan yang tinggi karena sistem ini menjadi pusat autentikasi bagi layanan. Ancaman keamanan autentikasi seperti *brute force* dan *credential stuffing* merupakan ancaman nyata untuk SSO [3]. *Brute force* merupakan serangan dengan mencoba berbagai kombinasi kredensial secara berulang untuk mendapatkan *password* yang benar [4], sedangkan *credential stuffing* merupakan serangan otomatis yang memanfaatkan pasangan *username* dan *password* yang telah diperoleh dari sumber lain [5].

Oleh sebab itu, analisis log menjadi salah satu hal yang dapat digunakan untuk melakukan deteksi keamanan. Penelitian menunjukkan bahwa analisis log bisa digunakan untuk mendeteksi aktivitas mencurigakan yang terjadi pada sistem melalui analisis perilaku login [6][7][8]. Dengan demikian analisis log memungkinkan sistem untuk melakukan deteksi dini terhadap aktivitas mencurigakan yang berpotensi mengganggu keamanan siber. Lebih lanjut, pengelolaan hasil deteksi tersebut memerlukan platform *Security Incident Response* yang handal seperti TheHive untuk sentralisasi penanganan kasus [9]. Namun, tantangan utama yang sering muncul adalah ketika sistem deteksi langsung diintegrasikan ke platform manajemen insiden tanpa

e-ISSN: 2548-6861

penyaringan. Pada kasus serangan masif, hal ini akan memicu penumpukan peringatan duplikat yang membebani analis keamanan dan memperlambat waktu tanggap insiden.

Berdasarkan uraian latar belakang, rumusan masalah dalam penelitian ini mencakup beberapa hal yaitu pertama, bagaimana mengidentifikasi aktivitas mencurigakan berdasarkan log *login error* pada sistem *Single Sign-On* (SSO). Kedua, bagaimana merancang dan mengimplementasikan sistem deteksi insiden keamanan berbasis analisis log *login* menggunakan Keycloak. serta ketiga, bagaimana mengintegrasikan hasil deteksi tersebut ke dalam sistem manajemen insiden yang mencakup pengelolaan data, deduplikasi *alert*, dan otomatisasi penanganan insiden menggunakan Redis, TheHive, dan n8n.

II. METODE

A. Jenis, Sifat, dan Pendekatan Penelitian

Jenis penelitian yang digunakan adalah Penelitian dan Pengembangan (*Research and Development* / R&D) [10]. Pendekatan ini dipilih karena penelitian secara khusus ditujukan untuk menganalisis celah keamanan pada sistem yang berjalan, lalu merancang dan mengimplementasikan produk solusinya. Produk yang dikembangkan berupa arsitektur sistem deteksi dan manajemen insiden keamanan siber berbasis analisis log autentikasi *Single Sign-On* (SSO) terintegrasi menggunakan Keycloak, Redis, TheHive, dan n8n.

Untuk membuktikan keandalan dan keefektifan produk yang telah dibangun, penelitian ini menerapkan metode pengujian kuantitatif eksperimental [11]. Evaluasi kinerja sistem dilakukan melalui eksperimen terkontrol dengan menyuntikkan simulasi serangan secara langsung ke dalam lingkungan uji. Hasil respons dari sistem kemudian diukur dan dihitung secara numerik berdasarkan metrik performa yang telah ditetapkan, seperti keberhasilan dalam mendeteksi anomali, waktu pemrosesan *end-to-end*, dan tingkat keberhasilan deduplikasi *alert*.

B. Subjek dan Teknik Pengumpulan Data

Subjek utama dalam penelitian ini mencakup aliran log autentikasi pengguna pada SSO Keycloak, secara spesifik pada *event* LOGIN_ERROR, serta data siklus hidup *alert* yang melintasi Redis, n8n, dan TheHive. Teknik pengumpulan data dilakukan melalui metode observasi *real-time* dan simulasi pengujian secara langsung (*live monitoring*), yang terbagi ke dalam dua fokus pencatatan data:

1. Pengumpulan Data Deteksi Ancaman**:** Dilakukan dengan menyimulasikan serangan secara langsung untuk memicu *event* LOGIN_ERROR. Data yang dikumpulkan meliputi kemampuan *worker* dalam membaca log secara terus-menerus (*stream reading*), mengekstrak *field* esensial (waktu kejadian, *username*, alamat IP, pesan *error*), dan mencocokkannya dengan aturan (*rule*) keamanan tanpa menunggu akumulasi log historis.
2. Pengumpulan Data Manajemen *Alert* dan Deduplikasi**:** Dilakukan dengan mengamati dan mencatat siklus lalu lintas *alert*. Data yang direkam meliputi keberhasilan pembuatan *dedup key* di dalam Redis, respons pengecekan data di dalam *dedup_list* (mekanisme deduplikasi), hingga rekaman eksekusi logika (*execution log*) pada n8n yang menentukan apakah sistem harus membuat tiket insiden (*case*) baru atau sekadar memperbarui (*update*) tiket yang sudah ada di TheHive.

C. Metode Analisis Data

Data yang terkumpul dari hasil pengujian dianalisis menggunakan metode deskriptif kuantitatif. Evaluasi dilakukan dengan membandingkan respons sistem terhadap parameter keberhasilan atau ambang batas (*threshold*) yang telah ditetapkan.

Nilai ambang batas deteksi *brute force* (7 kegagalan dalam 30 detik) dan *credential stuffing* (5 *username* berbeda dalam 120 detik dari 1 IP) ditetapkan berdasarkan pendekatan heuristik serta hasil peninjauan lapangan di PT. XYZ.. Penetapan angka ini tetap merujuk pada kerangka konseptual yang direkomendasikan oleh *National Institute of Standards and Technology* (NIST) SP 800-63B terkait praktik pembatasan upaya *login* (*rate limiting*) [12], serta prinsip *smart lockout* yang diterapkan pada penyedia layanan IAM seperti Microsoft Entra ID sebagai pembanding praktik industri [13]. Untuk deteksi *Credential Stuffing*, penentuan parameter turut mempertimbangkan definisi konseptual *Open Web Application Security Project* (OWASP) OAT-008 mengenai pola anomali rasio target akun terhadap satu sumber IP [14], meskipun OWASP tidak menetapkan nilai ambang batas numerik secara spesifik. Dengan demikian, referensi-referensi tersebut berfungsi sebagai landasan konseptual dari penentuan tentang pentingnya ambang batas kegagalan untuk penilaian terhadap kemungkinan insiden.

Untuk operasional sistem terkait potensi *false positive* yaitu risiko pengguna sah yang salah memasukkan *password* secara berulang dikenali sebagai serangan, penelitian ini melakukan analisis tambahan terhadap logika algoritma deteksi yang ada terhadap perilaku yang anggap sebagai *legitimate user.* Untuk menghindari asumsi arbitrer terkait kecepatan interaksi pengguna saat melakukan percobaan ulang (*retry*), jeda antar percobaan diturunkan menggunakan *Keystroke-Level Model* (KLM), metode standar dalam bidang *Human-Computer Interaction* (HCI) untuk memprediksi durasi tugas rutin pengguna berdasarkan penjumlahan waktu operator dasar yang telah divalidasi secara empiris [15].

Tugas "mencoba ulang *login* setelah gagal" diuraikan menjadi operator: **M** (persiapan mental, 1,35 detik), **H**

(memindahkan tangan ke *keyboard*, 0,40 detik), **K** (menekan tombol, 0,12–1,20 detik/karakter tergantung kecepatan mengetik), dan **B** (menekan tombol *submit*, 0,10–0,20 detik). Dengan asumsi panjang *password* delapan karakter, diperoleh estimasi :

- Tercepat (*typist* mahir, 90 wpm, tangan sudah di *keyboard*): 1,35 + 8(0,12) + 0,10 = 2,41 detik
- Tipikal (*typist* rata-rata, 40 wpm): 1,35 + 0,40 + 8(0,28) + 0,20 = 4,19 detik
- Terlambat (*hunt-and-peck*): 1,35 + 0,40 + 8(1,20) + 0,20 = 11,55 detik

Rentang 2,41–11,55 detik ini yang dipakai sebagai batas distribusi untuk melakukan pengujian false positive.

Kinerja arsitektur *end-to-end* dievaluasi berdasarkan empat rumusan metrik utama:

1. *Brute Force Detection*: Mengukur kemampuan sistem dalam mengidentifikasi pola serangan brute force berdasarkan jumlah kegagalan login dari alamat IP yang sama dalam rentang waktu tertentu. Sistem dinyatakan berhasil mendeteksi jika akumulasi kegagalan memenuhi kondisi matematis:

$$\sum_{i=1}^{n} Login\_Fail_{IP} = 7 \text{ untuk } \Delta t \leq 30s$$

2. *Credential Stuffing Detection*: Mengukur kemampuan sistem dalam mengidentifikasi pola credential stuffing berdasarkan jumlah username unik yang gagal login dari satu alamat IP dalam rentang waktu tertentu. Sistem memicu *alert* jika jumlah *username* unik yang gagal login dari satu IP memenuhi:

$$\sum_{i=1}^{n} Unique\_user_{IP} = 5 \text{ untuk } \Delta t \leq 120s$$

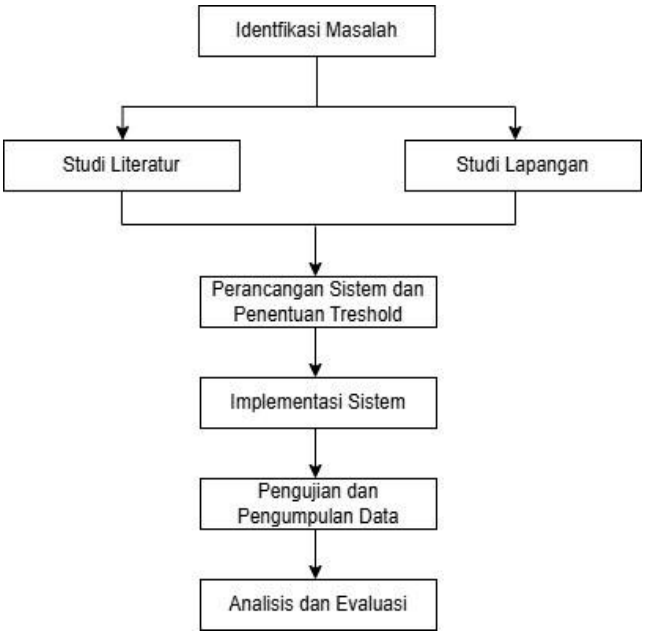
3. *False Positive Rate*: Mengukur potensi sistem keliru mengklasifikasikan trafik *login* gagal dari pengguna sah sebagai serangan. Tingkat *false positive* dihitung dengan membandingkan jumlah *alert* yang terbentuk dari trafik *legitimate* terhadap jumlah *alert* yang diharapkan apabila volume trafik yang sama berasal dari serangan sungguhan.

$$FPR = \frac{\sum_{i=1}^{k} Alert_{legitimate,i}}{\sum_{i=1}^{k} Alert_{expected,i}} x\ 100$$

4. *Alert Readiness for Deduplication*: Mengevaluasi efektivitas mekanisme penyaringan data pada pangkalan data Redis dalam mencegah masuknya peringatan duplikat ke dalam platform TheHive. Sistem dinyatakan berhasil secara absolut apabila mampu mencegat seluruh peringatan susulan, sehingga memastikan hanya terbentuk tepat satu tiket insiden unik untuk setiap entitas penyerang, terlepas dari jumlah *alert* yang dihasilkan oleh sensor padahari tersebut

5. *End-to-End Processing Time*: Menganalisis efisiensi otomatisasi respons insiden. Metrik ini mengukur total jeda waktu sejak log *error* pertama kali terbaca hingga *alert/case* berhasil dibuat di TheHive:

$$T_{Proses} = T_{TheHive} - T_{alert\_terdeteksi}$$



Gambar 1. Alur Penelitian

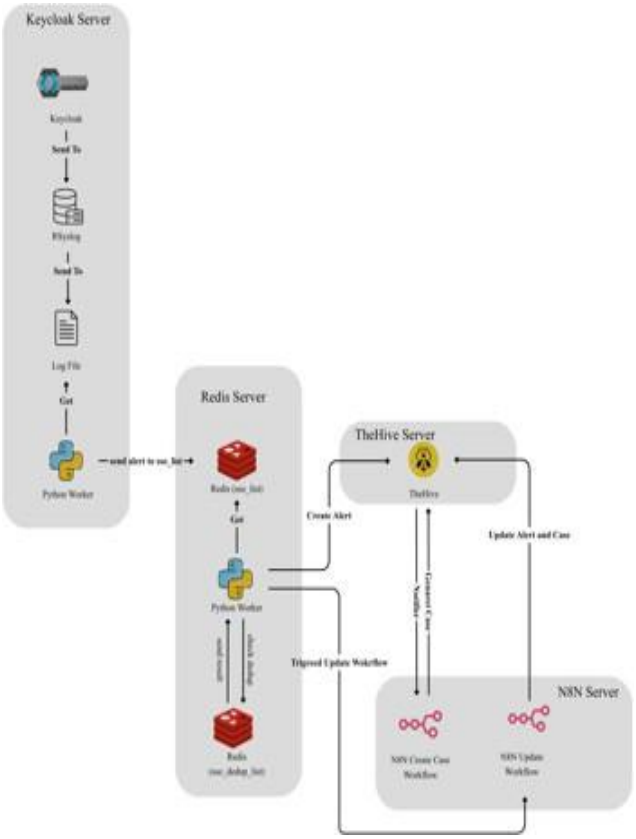
e-ISSN: 2548-6861

Penelitian ini dilaksanakan melalui tahapan yang sistematis untuk memastikan seluruh tujuan penelitian tercapai dengan baik. Alur tahapan penelitian (sebagaimana diilustrasikan pada Gambar 1) dijabarkan sebagai berikut:

Tabel 1. Deskripsi Alur Penelitian

Tahapan Penelitian	Penjelasan
Identifikasi Masalah	Merupakan tahap awal untuk merumuskan urgensi penelitian.
Studi Pustaka dan Studi Lapangan	Dilakukan secara paralel. Studi pustaka menghimpun landasan teori mengenai sistem. Studi lapangan dilakukan dengan mengobservasi kondisi aktual lalu lintas log autentikasi pengguna pada sistem operasi yang berjalan dan juga melihat kapasitas sumber daya yang bisa digunakan.
Perancangan Sistem dan Penentuan Threshold	Merancang topologi arsitektur sistem deteksi dan manajemen insiden yang terintegrasi.
Implementasi Sistem	Melakukan instalasi dan konfigurasi seluruh komponen (Keycloak, Redis, TheHive, n8n) ke dalam lingkungan uji (*testbed*).
Pengujian dan Pengumpulan Data	Mengeksekusi pengujian dengan menyuntikkan skenario simulasi serangan buatan secara langsung ke sistem. pembuatan tiket insiden di TheHive.
Analisis dan Evaluasi	Mengolah data hasil pengujian secara numerik. Evaluasi dilakukan dengan menghitung metrik performa untuk membuktikan keandalan dan latensi sistem.

E. Design dan Arsitektur Sistem



Gambar 2. Arsitektur Sistem

1. Arsitektur Sistem

Sistem yang dirancang dalam penelitian ini merupakan arsitektur terintegrasi yang menghubungkan beberapa komponen utama yang terdistribusi didalam 4 server.

Komponen pertama adalah *server* Keycloak yang terdiri dari beberapa elemen yang bekerja secara berurutan. Keycloak berperan sebagai aplikasi *Single Sign-On (sso)* yang mengelola seluruh proses autentikasi pada sistem, serta mengirimkan log login error ke layanan RSyslog yang kemudian meneruskannya ke dalam log file. Data inilah yang menjadi sumber data utama yang dibaca oleh *worker* python yang berfungsi sebagai sensor, untuk kemudian akan menghasilkan *alert* jika ditemukan perilaku mencurigakan dan meneruskan *alert* tersebut ke dalam Redis.

Selanjutnya, *server* Redis memegang peranan penting dalam deduplikasi pada arsitektur ini. Pada server ini terdapat aplikasi Redis yang menjadi tempat penyimpanan

sementara seluruh *alert* yang sudah pernah dikirmkan menuju TheHive pada hari tersebut. Di dalam *server* ini juga terdapat sebuah Python *worker* yang berfungsi sebagai mesin deduplikasi. *Worker* ini memeriksa keberadaan dedup key pada sso_dedup_list di Redis. Apabila dedup key ditemukan, worker mencari ID *alert* (hive_id) yang sesuai di TheHive dan meneruskannya ke n8n untuk memperbarui *alert* dan case yang sudah ada. Apabila dedup key tidak ditemukan, worker langsung membuat *alert* baru di TheHive melalui pemanggilan API. N8n server adalah server khusus yang dibuat untuk penggunaan n8n *automation tools.* N8n hadir dengan 2 workflow utama yang berfokus pada proses promote yaitu proses yang berfungsi mempromosikan *alert* menjadi sebuah case setelah menerima notifikasi dari TheHive dan proses update yaitu melakukan update baik pada case dan *alert* terkait hal-hal yang dibutuhkan.

Dan yang terakhir adalah Thehive server dimana pada server ini terdapat aplikasi yang menjadi ujung pipeline yaitu pengelolaan dan pencatatan insiden keamanan siber secara terpusat.

2. Mekanisme Deteksi Ancaman

Alur kerja sistem dimulai dari Keycloak *Server*. Setiap kali terjadi kegagalan *login*, Keycloak mencatat event *LOGIN_ERROR* dan mengirimkannya ke RSyslog. RSyslog kemudian meneruskan data log tersebut ke dalam sebuah Log File yang tersimpan secara lokal di Keycloak *Server*. Python *worker* pada sisi Keycloak membaca *Log File* tersebut secara terus-menerus, mengekstrak field-field esensial meliputi timestamp, username, alamat IP sumber, dan pesan error. Data yang diekstrak kemudian dibandingkan dengan aturan keamanan yang sudah didefinisikan untuk memeriksa apakah akumulasi kegagalan tersebut memenuhi batas serangan baik itu *brute force* maupun *credential stuffing*. Apabila memenuhi salah satu aturan tersebut, maka sensor menghasilkan sebuah *alert* beserta dedup *key* yang kemudian dikirimkan menuju *server* Redis.

3. Mekanisme Deduplikasi

Setelah *alert* terpicu, Python Worker pada Redis Server melakukan pengecekan dedup *key* pada sso_dedup_list untuk mengetahui apakah dedup *key* yang terbentuk sudah pernah ada sebelumnya. Hasil pengecekan ini (send result) menentukan percabangan alur yang akan dieksekusi selanjutnya.

Apabila *dedup key* tidak ditemukan di dalam Redis, *worker* Python mengklasifikasikannya sebagai ancaman baru dan melakukan panggilan API secara langsung ke TheHive untuk membentuk *alert* baru. Setelah *alert* berhasil dibuat, TheHive secara otomatis mengirimkan notifikasi ke n8n melalui mekanisme *notifier* yang telah dikonfigurasi sebelumnya. Notifikasi ini memicu n8n untuk menjalankan workflow Create Case, yang bertugas mempromosikan *alert* tersebut menjadi sebuah tiket insiden (*case*) secara otomatis. Pada saat yang bersamaan, *dedup key* tersebut dituliskan ke dalam *cache* Redis menggunakan parameter SET EX dengan masa kedaluwarsa selama 86.400 detik (24 jam). Melalui penyisipan format tanggal pada sintaks kunci tersebut, mekanisme ini akan memblokir pembuatan tiket baru untuk jenis serangan dan alamat IP yang sama hingga bergantinya hari kalender. Parameter TTL di sini berfungsi murni sebagai pembersihan memori otomatis (*garbage collection*) untuk menjaga efisiensi RAM Redis.

Apabila *dedup key* sudah eksis di dalam Redis, hal ini mengindikasikan adanya serangan lanjutan dari aktor yang sama pada hari tersebut. Pada kondisi ini, *worker* Python tidak akan membuat tiket baru, melainkan mengambil ID tiket (*hive_id*) asli di TheHive dan mengirimkan *id* tersebut menuju *Update Workflow* di *server* n8n. Platform n8n kemudian memperbarui *alert* yang sudah ada beserta *case* yang berasosiasi dengannya dengan menambahkan metrik atau observabel serangan terbaru.

III. HASIL DAN PEMBAHASAN

Bab ini menguraikan hasil pengujian dan evaluasi terhadap arsitektur sistem deteksi dan manajemen insiden keamanan siber berbasis analisis log *Single Sign-On* (SSO) Keycloak. Evaluasi kinerja sistem diukur secara kuantitatif berdasarkan empat metrik utama, yaitu akurasi deteksi *brute force*, akurasi deteksi *credential stuffing*, tingkat keberhasilan deduplikasi, serta waktu pemrosesan *end-to-end*.

A. Lingkungan dan Skenario Pengujian

Pengujian performa sistem dilakukan melalui simulasi serangan secara terdistribusi di dalam lingkungan uji (*testbed*). Proses injeksi log anomali dilakukan menggunakan skrip otomatisasi (cURL) yang menembakkan permintaan autentikasi gagal secara *real-time* ke titik akhir (*endpoint*) Keycloak.

Untuk menguji ketahanan sistem dalam menangani beban kerja setiap skenario pengujian (baik *Brute Force* maupun *Credential Stuffing*) dilakukan dengan menyuntikkan lalu lintas serangan dari dua alamat IP penyerang secara bersamaan. Dalam setiap iterasi, alamat IP x.188 dan x.210

e-ISSN: 2548-6861

secara simultan mengirimkan permintaan *login* gagal ke akun yang sama dengan target ambang batas deteksi yang setara. Metode pengujian simultan ini dipilih untuk membuktikan bahwa sensor deteksi mampu menjaga konsistensi variabel hitungan per entitas secara terpisah meskipun terdapat aliran data yang tumpang tindih (*overlapping log streams*). Dengan cara ini, keandalan arsitektur dalam melakukan *state flushing* dan deduplikasi peringatan dapat diuji pada kondisi beban kerja yang dinamis dan kompetitif.

B. Hasil Pengujian dan Pembahasan

1. *Brute Force Detection*

Tabel 2. Hasil Pengujian *Brute Force Detection*

Simulasi	IP	Total Login Error	*Alert* Terbentuk	MTTD
BF1	x.188	100	14	0.297 s
BF1	x.210	100	14	0.615 s
BF2	x.188	100	14	0.539 s
BF2	x.210	100	14	0.678 s
BF3	x.188	100	14	0.514 s
BF3	x.210	100	14	0.490 s
BF4	x.188	100	14	0.527 s
BF4	x.210	100	14	0.440 s

Berdasarkan Tabel 2, seluruh skenario brute force menghasilkan 14 *alert* dari 100 login gagal pada masing-masing IP. Hasil ini sesuai dengan threshold sistem, yaitu 7 kegagalan login dalam 30 detik. Seluruh *alert* terbentuk secara konsisten pada keempat simulasi, dengan rata-rata waktu deteksi 0,513 detik setelah kegagalan login ketujuh terjadi.

2. *Credential Stuffing*

Tabel 3. Hasil Pengujian *Credential Stuffing*

Simulasi	IP	Total Login Error	*Alert* Terbentuk	MTTD
CS1	x.188	100	20	0.561 s
CS1	x.210	100	20	0.532 s
CS2	x.188	100	20	0.574 s
CS2	x.210	100	20	0.583 s
CS3	x.188	100	20	0.554 s
CS3	x.210	100	20	0.537 s
CS4	x.188	100	20	0.568 s
CS4	x.210	100	20	0.709 s

Berdasarkan Tabel 3, seluruh skenario *credential stuffing* berhasil terdeteksi oleh sistem. Pada setiap simulasi, 100 login gagal dari masing-masing IP menghasilkan 20 *alert*, sesuai ambang batas lima username unik dalam 120 detik. Hasil ini menunjukkan sensor mampu membedakan *credential stuffing* dari *brute force* dan bekerja secara konsisten tanpa kegagalan deteksi, dengan rata-rata waktu deteksi sebesar 0,577 detik.

3. *False Positive Rate*

Tabel 4. *Brute Force False Positive rate*

Simulasi	IP	Legitimate Error	Expected	*Alert* Create
BF-FP1	x.188	2000	285	29
BF-FP1	x.210	2000	285	23
BF-FP2	x.188	2000	285	37
BF-FP2	x.210	2000	285	24
BF-FP3	x.188	2000	285	26
BF-FP3	x.210	2000	285	31
BF-FP3	x.188	2000	285	29
BF-FP3	x.210	2000	285	29

Tabel 5. *Credential Stuffing False Positive rate*

Simulasi	IP	Legitimate Error	Expected	*Alert* Create
CS-FP1	x.188	2000	400	22
CS-FP1	x.210	2000	400	23
CS-FP2	x.188	2000	400	18
CS-FP2	x.210	2000	400	13
CS-FP3	x.188	2000	400	16
CS-FP3	x.210	2000	400	15
CS-FP4	x.188	2000	400	17
CS-FP4	x.210	2000	400	14

Berdasarkan Tabel 4 dan Tabel 5, Rata-rata tingkat *false positive* pada skenario *brute force* sekitar 10,18%, lebih dari dua kali lipat dibandingkan skenario *credential stuffing* yang hanya sekitar 4,31%. Hal ini mengindikasikan bahwa ambang batas *brute force* (7 kegagalan/30 detik) relatif lebih rentan terhadap *false positive* dibandingkan ambang batas *credential stuffing* (5 *username*/120 detik) pada kondisi trafik pengguna sah dengan volume kegagalan tinggi.

4. *Deduplication*

Tabel 6. Metrik Keberhasilan Deduplikasi *Brute Force*

Simulasi	IP	*Alert* Terbentuk	*Alert* Baru Thehive	*Alert* Update	Dedup Status

BF1	x.188	14	1	13	✅
BF1	x.210	14	1	13	✅
BF2	x.188	14	1	13	✅
BF2	x.210	14	1	13	✅
BF3	x.188	14	1	13	✅
BF3	x.210	14	1	13	✅
BF4	x.188	14	1	13	✅
BF4	x.210	14	1	13	✅

Tabel 7. Metrik Keberhasilan Deduplikasi *Credential Stuffing*

Simulasi	IP	*Alert* Terbentuk	*Alert* Baru Thehive	*Alert* Update	Dedup Status
CS1	x.188	20	1	19	✅
CS1	x.210	20	1	19	✅
CS2	x.188	20	1	19	✅
CS2	x.210	20	1	19	✅
CS3	x.188	20	1	19	✅
CS3	x.210	20	1	19	✅
CS4	x.188	20	1	19	✅
CS4	x.210	20	1	19	✅

Berdasarkan Tabel 6 dan Tabel 7, mekanisme deduplikasi pada Redis dan TheHive berjalan dengan baik pada seluruh skenario pengujian. Setiap entitas penyerang hanya menghasilkan satu *alert* baru di TheHive, sedangkan *alert* berikutnya diarahkan sebagai update pada tiket yang sudah ada. Hasil ini menunjukkan bahwa sistem mampu mencegah terbentuknya tiket insiden duplikat, baik pada skenario brute force maupun credential stuffing, sehingga deduplikasi berjalan konsisten sesuai rancangan.

5. *End-to-End Processing Time*

Tabel 8. *End-to-End Processing*

Simulasi	IP	Waktu Alert Terdeteksi	Waktu Alert Terbentuk Pada TheHive	Δ T alert terbentuk
BF1	x.188	14:41:17,286	14:41:19,652	2,366 s
BF1	x.210	14:41:19,343	14:41:21,678	2,335 s
CS1	x.188	09:39:08,018	09:39:10,535	2,517 s
CS1	x.210	09:40:56,507	09:40:59,149	2,642 s
BF2	x.188	14:51:58,346	14:52:00,921	2,575 s
BF2	x.210	14:51:55,292	14:51:57,905	2,613 s
CS2	x.188	11:55:14,058	11:55:18,113	4,055 s
CS2	x.210	11:55:14,032	11:55:17,100	4,055 s
BF3	x.188	15:08:37,650	15:08:40,361	2,711 s
BF3	x.210	15:08:39,669	15:08:42,372	2,703 s
CS3	x.188	14:02:15,905	14:02:19,365	3,460 s
CS3	x.210	14:02:05,752	14:02:09,280	3,528 s
BF4	x.188	15:21:08,808	15:21:11,193	2,385 s
BF4	x.210	15:21:08,830	15:21:12,207	3,377 s
CS4	x.188	16:26:31,030	16:26:34,372	3,342 s
CS4	x.210	16:26:22,908	16:26:26,217	3,309 s

Berdasarkan hasil pengukuran *end-to-end* (n=16), waktu dari *alert* terdeteksi hingga *case* terbentuk di TheHive berada pada rentang 2,335–4,055 detik, dengan rata-rata 2,998 detik (SD = 0,586 detik; 95% CI: 2,686–3,311 detik). Nilai standar deviasi yang relatif kecil dibanding rata-ratanya mengindikasikan bahwa waktu pemrosesan *end-to-end* tidak menyebar jauh dari nilai tengahnya.

6. Analisis Keamanan dan Resiliensi Komponen Pipeline

Selain evaluasi kinerja deteksi, penelitian ini turut menganalisis dua aspek arsitektural yang menjadi perhatian pada sistem *production-grade*: keamanan komponen *pipeline* itu sendiri, serta resiliensinya terhadap kegagalan masing-masing komponen.

a. Keamanan Komponen Pipeline

Keamanan *pipeline* itu sendiri turut menjadi pertimbangan penting, mengingat sistem deteksi dan manajemen insiden ini juga merupakan bagian dari permukaan serangan infrastruktur PT. XYZ. Redis pada implementasi ini telah menggunakan autentikasi, namun komunikasi antar komponen (*worker*, Redis, TheHive, n8n) belum sepenuhnya menerapkan TLS/HTTPS secara konsisten, dan keempat server masih berada pada satu jaringan tanpa segmentasi. Kondisi ini menimbulkan dua risiko:

(1) komunikasi tanpa TLS berpotensi disadap atau dimanipulasi jika penyerang memperoleh akses jaringan internal

(2) tanpa segmentasi jaringan, kompromi pada satu komponen berpotensi memungkinkan pergerakan lateral ke komponen lain, termasuk TheHive yang menyimpan seluruh data insiden.

b. Resiliensi Terhadap Kegagalan Komponen

Pipeline terdiri dari dua *python worker* yang berjalan terpisah yaitu Worker Deteksi di server Keycloak dan Worker Dedup & *Alert* Creation di server Redis yang masing-masing bergantung pada ketersediaan Redis, TheHive, dan n8n. Tabel 9 merangkum dampak operasional apabila salah satu komponen mengalami kegagalan

e-ISSN: 2548-6861

Tabel 9. Analisis Dampak Kegagalan Komponen

Komponen	Dampak Operasional
Worker Deteksi (server Keycloak)	Tidak ada log baru yang dibaca/dianalisis; deteksi berhenti total
Worker Dedup & *Alert* Creation (server Redis)	*Alert* yang sudah terdeteksi tidak dapat diproses sama sekali
TheHive	Berisiko hilang jika Worker Dedup *restart* sebelum TheHive pulih
Redis data store	Berisiko hilang jika *worker restart* selama Redis *down*
n8n	Insiden tidak terkelola secara terstruktur

IV. KESIMPULAN

A. Kesimpulan Penelitian

Penelitian ini merancang dan mengimplementasikan sistem deteksi dan manajemen insiden keamanan siber berbasis analisis log autentikasi SSO Keycloak, terintegrasi dengan Redis, TheHive, dan n8n. Pada seluruh skenario pengujian yang dilakukan, sistem berhasil mengidentifikasi seluruh pola serangan yang disimulasikan, dengan tingkat keberhasilan deteksi 100% untuk kedua jenis serangan, dan rata-rata waktu pembentukan *alert* sebesar 0,513 detik (BF) dan 0,577 detik (CS). Namun demikian, hasil simulasi false positive menunjukkan bahwa ambang batas yang digunakan tetap berpotensi menghasilkan peringatan keliru pada kondisi trafik pengguna sah bervolume tinggi, dengan tingkat false positive (FPR) sebesar 10,18% untuk brute force dan 4,31% untuk credential stuffing. Temuan ini menunjukkan bahwa sistem memiliki kemampuan deteksi yang kuat, namun belum sepenuhnya bebas dari risiko kesalahan klasifikasi, terutama pada topologi jaringan shared IP/NAT.

Mekanisme deduplikasi berbasis Redis berjalan efektif pada seluruh skenario yang diuji dimana setiap rentetan *alert* hanya menghasilkan satu tiket insiden unik di TheHive per entitas per hari, sementara *alert* susulan diarahkan sebagai pembaruan melalui *workflow* n8n. Mekanisme ini turut membatasi dampak operasional dari *false positive* yang teridentifikasi, menjadi maksimal satu *case* keliru per hari per entitas.

Dari sisi efisiensi pemrosesan, rata-rata waktu end-to-end dari deteksi *alert* hingga pembentukan case di TheHive adalah 2,937 detik (rentang 2,335–4,055 detik), menunjukkan alur otomatisasi berjalan responsif pada kondisi pengujian yang diterapkan. Secara keseluruhan, arsitektur yang dibangun menunjukkan potensi yang baik dalam mendukung deteksi dan pengelolaan insiden keamanan pada lingkungan SSO, meskipun hasil ini perlu dimaknai dalam konteks lingkup dan keterbatasan pengujian yang dijelaskan pada subbagian berikut.

B. Saran dan Keterbatasan Penelitian

Penelitian ini memiliki beberapa keterbatasan yang perlu menjadi perhatian dalam menginterpretasikan hasil, sekaligus arah pengembangan lanjutan:

1. Variasi IP dan skenario pengujian yang terbatas.
   Pengujian, baik untuk deteksi serangan maupun simulasi. *false positive*, menggunakan jumlah alamat IP dan variasi skenario yang masih terbatas. Pengujian lanjutan dengan lebih banyak variasi IP, pola serangan, serta karakteristik trafik pengguna sah diperlukan untuk memperkuat generalisasi hasil penelitian ini

2. Ruang pengembangan arsitektur lebih lanjut**.** Sebagaimana diuraikan pada bagian Analisis Keamanan dan Resiliensi Komponen *Pipeline*, arsitektur saat ini masih memiliki ruang perbaikan, antara lain penerapan TLS secara menyeluruh, segmentasi jaringan antar komponen, mekanisme antrian persisten untuk menjamin durabilitas data saat komponen mengalami gangguan, serta skema deteksi yang mempertimbangkan kombinasi IP dan *username* guna menekan tingkat *false positive* pada topologi jaringan *shared IP*

DAFTAR PUSTAKA

[1] A. Pashalidis, C. J. Mitchell, and R. Holloway, “A Taxonomy of *Single Sign-On* Systems.” https://doi.org/10.1007/3-540-45067-X_22

[2] T. Bazaz and A. Khalique, “A Review on Single Sign on Enabling Technologies and Protocols,” 2016. https://doi.org/10.5120/ijca2016911938

[3] A. Zineddine, Y. Belfaik, A. Rehaimi, Y. Sadqi, and S. Safi, “*Single Sign-On* Security and Privacy: A Systematic Literature Review” https://doi.org/10.32604/cmc.2025.066139

[4] “*Brute force* Attack | OWASP Foundation.” https://owasp.org/www-community/attacks/Brute_force_attack

[5] “*Credential stuffing* | OWASP Foundation.” https://owasp.org/www-community/attacks/Credential_stuffing

[6] S. S. Rajan, “Data-driven Security Analysis of System Audit Logs for Intrusion Detection and Prevention,” *International Journal of Computer Science & Security (IJCSS)*, no. 19, p. 136. https://doi.org/10.13140/RG.2.2.21377.93284

[7] U. K. Raut, “Log Based Intrusion Detection System,” vol. 20, no. 5, pp. 15–22, https://doi.org/10.9790/0661-2005011522

[8] L. Wang and M. K. Reiter, "Detecting Stuffing of a User’s Credentials at Her Own Accounts," *NDSS*, 2020. https://doi.org/10.48550/arXiv.1912.11118

[9] R. Groenewegen and M. Janssen, "TheHive Project: The maturity of an open-source Security Incident Response platform," 2021. https://www.researchgate.net/publication/352715439_TheHive_Project_The_maturity_of_an_open-source_Security_Incident_Response_platform

[10] Sugiyono, *Metode Penelitian Kuantitatif, Kualitatif, dan R&D*. Bandung: Alfabeta, 2019. https://www.researchgate.net/publication/377469385_METODE_PENELITIAN_KUANTITATIF_KUALITATIF_DAN_RD

[11] J. W. Creswell and J. D. Creswell, *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*, 5th ed. Los Angeles: SAGE Publications, 2018. https://www.ucg.ac.me/skladiste/blog_609332/objava_105202/fajlovi/Creswell.pdf

[12] P. A. Grassi, M. E. Garcia, and J. L. Fenton, "Digital Identity Guidelines: Authentication and Lifecycle Management," National Institute of Standards and Technology (NIST), Special Publication 800-63B, Jun. 2017. https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-63b.pdf

[13] Microsoft, "Protect user accounts from attacks with Microsoft Entra smart lockout," Microsoft Entra Documentation.

https://learn.microsoft.com/en-us/entra/identity/authentication/howto-password-smart-lockout

[14] OWASP Foundation, "OAT-008 Credential Stuffing," OWASP Automated Threat Handbook. https://owasp.org/www-project-automated-threats-to-web-applications/assets/oats/EN/OAT-008_Credential_Stuffing

[15] Card, S. K., Moran, T. P., & Newell, A. (1980). "The keystroke-level model for user performance time with interactive systems." *Communications of the ACM*, 23(7), 396–410. https://doi.org/10.1145/358886.358895