BAB 1
PENDAHULUAN

1.1 Latar Belakang Masalah

Proses pembaruan data pribadi karyawan di PT. XYZ masih dilakukan secara manual melalui *Human Resource* (HR) dan dinilai tidak efektif karena memiliki potensi keterlambatan, kesalahan perubahan, dan memiliki risiko kebocoran data. Undang-Undang Perlindungan Data Pribadi (UU PDP) menyatakan setiap proses pengelolaan data pribadi wajib dilakukan secara akurat, transparan, dan aman [1]. Untuk itu dibangun aplikasi internal *Employee Self Service* (ESS) agar karyawan dapat memperbarui data secara mandiri dan aman. Arsitektur yang digunakan aplikasi ESS adalah arsitektur terpisah (*decoupled architecture*) antara *back-end* dan *front-end*. *Back-end* menjadi *single source of truth* dan *authorization* peran berdasarkan *Role-Based Access Control* (RBAC) dengan layanan *Representational State Transfer Application Programming Interface* (REST API) [2]. Sedangkan pengembangan untuk sisi *website* dan *mobile* dikembangkan oleh tim *front-end* terpisah dan mengonsumsi *Application Programming Interface* (API) melalui *endpoint Representational State Transfer* (REST) yang telah dikembangkan oleh *back-end* sesuai dengan *OpenAPI Specification (OAS)*.

Kondisi saat ini, pengujian *endpoint* di PT. XYZ dilakukan secara manual menggunakan *Postman* di lingkungan *pre-production* dan lokal. Aplikasi ESS memiliki 56 *endpoint* dengan 28 di antaranya dikategorikan sebagai *critical endpoint* karena mengandung parameter identitas (*id*, *userId*, *resourceId*) yang berpotensi rentan terhadap serangan BAC. ESS menerapkan lima *permission* dan RBAC dengan dua peran utama yaitu *admin* dan karyawan. Pengujian manual yang dilakukan pada endpoint kritis menunjukkan beberapa keterbatasan seperti *coverage* pengujian hanya mencapai sekitar 50% dari total *endpoint* karena keterbatasan waktu dan sumber daya, waktu pengujian bervariasi signifikan antar *endpoint* (0,06 hingga 4,14 detik per *request*), dan tidak ada standarisasi proses pengujian yang menyebabkan inkonsistensi hasil. Dokumentasi hasil pengujian juga tidak terstruktur, menyulitkan *tracking* kerentanan dan proses *remediation*. Setiap *endpoint* rentan terhadap *Broken Access Control* (BAC) seperti *Insecure Direct Object Reference* (IDOR) [3] dan *Broken Object Level Authorization*

(BOLA) [4]. Pengujian manual membutuhkan waktu lama [5], rentan terlewat, dan sulit dikembangkan konsisten, sehingga *coverage* dan kecepatan deteksi BAC terbatas [6].

Penelitian ini mengembangkan BYEBAC atau *Broken Access Control Detector*, alat bantu berbasis AI *Agent* untuk otomatisasi pengujian *endpoint back-end* guna mendeteksi kerentanan BAC horizontal maupun vertikal pada aplikasi ESS di PT. XYZ dengan melakukan pengujian otomatis sekaligus memberikan ringkasan hasil pengujian.

Penelitian terkait menunjukkan pemanfaatan AI untuk otomatisasi pengujian keamanan. Analisis statis dengan *OpenAPI Specification* (OAS) dimanfaatkan untuk memetakan pola IDOR atau BOLA [7], sedangkan pengujian dinamis berbasis AI dieksplorasi untuk mendeteksi BAC vertikal secara *real-time* [8]. Namun, sebagian besar studi berfokus pada aplikasi *website* atau prototipe [9], atau menggunakan *Large Language Model* (LLM) pada *device Internet of Things* (IoT) [10]. Kesenjangan pengetahuan yang diangkat adalah penerapan AI *Agent* untuk otomatisasi pengujian *endpoint back-end* pada aplikasi internal masih terbatas.

Oleh karena itu, penelitian ini mengembangkan BYEBAC, sebuah alat bantu (*tools*) berbasis AI *Agent* untuk otomatisasi pengujian *endpoint back-end*. BYEBAC dirancang untuk mempelajari matriks RBAC, OAS, dan *policy rule endpoint* menggunakan *Large Language Model* (LLM) Gemini 3.0 *Pro Preview* sebagai *reasoning engine* untuk mendeteksi kerentanan *Broken Access Control*, baik *horizontal* maupun vertikal. AI *Agent* menghasilkan *test plan* otomatis dan menjalankan pengujian *HyperText Transfer Protocol* (HTTP) *Request* dengan berbagai *role* untuk validasi kontrol akses. Hasil pengujian berupa *artifacts JavaScript Object Notation* (JSON) serta *markdown* yang dilengkapi *AI-powered security summary* untuk dokumentasi evaluasi kerentanan.

Penelitian ini mengembangkan BYEBAC untuk meningkatkan keamanan aplikasi ESS melalui deteksi otomatis kerentanan BAC. Hasil pengujian pada 56 *endpoint* menunjukkan BYEBAC mampu mendeteksi kerentanan dengan akurasi 97,3% (*precision*: 91,4%, *recall*: 100%, *F1-score*: 95,5%), mengurangi waktu pengujian menjadi rata-rata dua menit per iterasi dengan *coverage* 100% *endpoint*, dibandingkan pengujian manual yang hanya mencakup sekitar 50% *endpoint* dengan waktu yang lebih lama sekitar tiga jam dan tiga detik dengan hasil tidak konsisten.

1.2 Rumusan Masalah

Berdasarkan masalah yang ada dan dihadapi, terdapat beberapa rumusan masalah yang hendak diselesaikan di antaranya:

1. Bagaimana merancang dan mengimplementasikan BYEBAC sebagai alat bantu berbasis AI *Agent* untuk pengujian otomatis kerentanan *Broken Access Control* (IDOR dan BOLA) pada *endpoint back-end* aplikasi ESS berbasis *OpenAPI Specification*, *policy rules*, dan matriks RBAC?
2. Bagaimana performa pengujian otomatis menggunakan BYEBAC dibandingkan dengan pengujian manual dari aspek kecepatan (*time to detect*), *coverage endpoint* dan *role*, dan akurasi deteksi (*precision*, *recall*, F1-*score*)?

1.3 Batasan Permasalahan

Dari rumusan masalah yang dijelaskan bagian sebelumnya, masalah dibatasi guna mempertajam penelitian untuk fokus dengan rumusan masalah yang dipaparkan sebelumnya berupa:

1. Ruang lingkup pengujian terbatas pada aplikasi internal *Employee Self Service* (ESS) dengan fokus utama pada 28 *endpoint* kritis dari total 56 *endpoint* yang mengandung parameter identitas (*id*, *userId*, *resourceId*) dan memerlukan validasi kontrol akses berbasis identitas pengguna.
2. Jenis kerentanan terbatas pada *horizontal* (IDOR) dan vertikal (BOLA). Celah keamanan lain seperti *Structured Query Language* (SQL) *injection*, *Cross Site Scripting* (XSS), dan *Cross Site Request Forgery* (CSRF) berada di luar cakupan.
3. *Role* mencakup *admin* dan *employee* dengan skenario uji menyesuaikan hak akses aktual (*permission*).
4. Sumber kebenaran akses didasarkan pada matriks RBAC yang berisi *mapping role* dengan *permission*, *policy rules*, dan *OpenAPI Specification*.
5. BYEBAC menggunakan *Large Language Model* (LLM) Gemini 3.0 *Pro Preview* sebagai *reasoning engine* untuk analisis kebijakan akses, generasi

test plan otomatis, dan pembuatan laporan keamanan dengan *AI-powered summary*.

6. Implementasi BYEBAC sebagai AI *Agent* berupa *orchestrator* yang memanfaatkan HTTP *Client* untuk eksekusi *request*, *Auth Manager* untuk *multi-role authentication* dengan *token caching*, serta evaluator untuk klasifikasi hasil pengujian berdasarkan *confusion matrix*.
7. Pengujian performa dilakukan dengan membandingkan hasil deteksi BYEBAC terhadap hasil pengujian manual sebagai *baseline*. BYEBAC dijalankan sebanyak 5 kali iterasi pada seluruh 28 *endpoint* untuk mengukur konsistensi dan stabilitas hasil. Sedangkan, Pengujian manual dilakukan pada *subset endpoint* kritis sebagai *ground truth* untuk validasi klasifikasi kerentanan. Hasil dari kedua pengujian akan dibandingkan dengan *confussion matrix* dan metrik evaluasi.
8. Hal di luar cakupan seperti perbaikan kode aplikasi (remediasi teknis), BYEBAC hanya menyediakan laporan deteksi kerentanan dengan AI-*powered security assessment* sebagai panduan evaluasi.

1.4 Tujuan Penelitian

Berdasarkan permasalahan yang ada, tujuan penelitian ini adalah:

1. Merancang dan membangun BYEBAC sebagai alat bantu berbasis AI *Agent* yang mampu menjalankan pengujian BAC dari segi *horizontal* (IDOR) dan vertikal (BOLA) pada *endpoint back-end* aplikasi ESS secara otomatis dengan *automated test plan generation*, *resource ID discovery*, dan *multi-role testing*.
2. Membandingkan performa pengujian otomatis dengan manual dari aspek kecepatan (*time to detect*, total *duration*), *coverage endpoint* dan *role*, dan evaluasi akurasi klasifikasi (*precision*, *recall*, F1-*score*) pada *subset* hasil yang divalidasi secara manual.
3. Mengevaluasi peningkatan efektivitas pengujian keamanan aplikasi ESS melalui perbandingan *coverage endpoint*, konsistensi hasil pengujian, dan efisiensi waktu antara metode manual dan BYEBAC.

1.5 Manfaat Penelitian

Penelitian yang dilakukan diharapkan dapat memberikan manfaat kepada beberapa pihak di antaranya:

1. **Manfaat Akademis**

   Penelitian ini berkontribusi dalam otomatisasi pengujian keamanan *endpoint back-end* menggunakan AI *Agent* yang memanfaatkan matriks RBAC dan OAS untuk mendeteksi IDOR dan BOLA. Penelitian ini menyajikan arsitektur *agent* berbasis siklus *plan–action–observe–evaluate* yang memungkinkan eksplorasi kerentanan secara sistematis dan menghasilkan temuan yang dapat direproduksi. Selain itu, penelitian ini memperkaya kajian ilmiah di bidang keamanan aplikasi dengan mengintegrasikan konsep *model-based testing*, pemetaan peran–*permission*, dan *LLM-assisted testing* dalam satu *pipeline* yang terukur melalui *confusion matrix* dan metrik evaluasi standar (precision, recall, F1-score, dan accuracy). Hasilnya dapat menjadi referensi bagi peneliti selanjutnya untuk mengembangkan kerangka kerja pengujian BAC berbasis AI pada domain, arsitektur sistem, maupun jenis kerentanan lain yang lebih luas.

2. **Manfaat Praktis**

   Penelitian ini memberikan manfaat praktis bagi organisasi dengan menghadirkan panduan konkret untuk mengimplementasikan *pipeline* pengujian keamanan *endpoint* yang efisien. BYEBAC sebagai alat bantu yang dikembangkan mampu meningkatkan *coverage* pengujian dari sekitar 50% secara manual menjadi 100% secara otomatis pada seluruh 56 *endpoint*, sekaligus mempercepat proses pengujian dengan durasi rata-rata sekitar dua menit per iterasi untuk 130 *test case*. Selain itu, BYEBAC menghasilkan deteksi yang akurat (akurasi 97,3% dengan *precision* 91,4%, *recall* 100%, dan *F1-score* 95,5%), menunjukkan konsistensi hasil melalui lima kali iterasi pengujian, serta menyediakan laporan terstruktur dengan *AI-powered security summary* yang membantu mempercepat proses *triage* dan *remediation*. Matriks RBAC dan *pipeline* otomatis ini dapat diintegrasikan ke dalam alur *Continuous Integration* dan *Continuous Delivery* (CI/CD) sehingga pengujian keamanan dapat berjalan secara berkelanjutan, menghemat waktu dan sumber daya, sekaligus meningkatkan tingkat keamanan aplikasi internal.

1.6 Sistematika Penulisan

Sistematika penulisan laporan adalah sebagai berikut:

- Bab 1 PENDAHULUAN
  Bab pertama membahas tentang latar belakang masalah penelitian, susunan rumusan masalah yang diambil dari masalah yang ada, batasan masalah dari suatu penelitian, tujuan, dan manfaatnya dilakukan penelitian untuk lingkungan sekitar. Selain itu juga membahas tentang sistematika penulisan.
- Bab 2 LANDASAN TEORI
  Bab kedua membahas tentang penelitian sebelumnya dan dijadikan referensi dilakukannya penelitian ini serta teori dan konsep yang akan digunakan untuk penelitian ini.
- Bab 3 METODOLOGI PENELITIAN
  Bab ketiga menjelaskan metode penelitian yang digunakan dalam penelitian ini serta perancangan dari solusi yang diusulkan.
- Bab 4 HASIL DAN DISKUSI
  Bab keempat menganalisis dan membahas hasil yang diperoleh dari solusi yang dilakukan.
- Bab 5 KESIMPULAN DAN SARAN
  Bab ini menjelaskan kesimpulan dan saran yang didapat dari solusi yang dilakukan dan penelitian apa yang dapat dilanjutkan dimasa yang akan datang.