JTIM : Jurnal Teknologi Informasi dan Multimedia

p-ISSN : 2715-2529
e-ISSN : 2684-9151
https://journal.sekawan-org.id/index.php/jtim

Analisis Keamanan *Website* Global Academic Information System menggunakan OWASP ZAP dan Model AI Lokal

Bayu Irfan Aditya [1], Asep Rio Saputra [2,*], Nova Teguh Sunggono [3], dan M. Bucci Ryando [4]

[1] Program Studi Teknik Informatika, Institut Teknologi dan Bisnis Bina Sarana Global, Indonesia

* Korespondensi: ars.asep123@gmail.com

Sitasi: Aditya, B. I.; Saputra, A. R.; Sunggono, N. T.; and Ryando, M. B. (**2025**). Analisis Keamanan Website Global Academic Infor-mation System menggunakan OWASP ZAP dan Model AI Lokal. JTIM: Jurnal Teknologi Informasi Dan Multimedia, 7(3), 490-503. https://doi.org/10.35746/jtim.v7i3.759

Diterima: 31-05-2025
Direvisi: 27-06-2025
Disetujui: 01-07-2025

Abstract: Academic websites serve as central platforms for managing higher education services, including academic records, financial data, and institutional communication. However, such systems are increasingly vulnerable to cyberattacks due to their internet exposure and insufficient protection against security flaws. This study proposes an integrated solution that combines automated scanning with OWASP ZAP and a local artificial intelligence model (Mistral) executed via the Ollama platform. The entire process is automated using Python scripting, covering stages such as spidering, active scanning, JSON result extraction, and AI-based mitigation recommendation generation. The research was conducted on the Global Academic Information System website. The scan results revealed a total of 193 vulnerabilities, including 4 high, 8 medium, 111 low, and 70 informational risks. Each vulnerability was analyzed using the local AI model to produce specific technical recommendations, such as adding security headers, implementing CSRF tokens, and configuring secure cookies. All outputs were automatically compiled into a structured Excel report suitable for developers. This approach proves effective in streamlining the security audit process, reducing manual workload, and preserving data privacy, as all operations are conducted locally without reliance on cloud services. The study demonstrates that integrating OWASP methods with local AI provides a practical, adaptive, and standalone solution for web application security testing.

Keywords: OWASP ZAP, Website Security, Local Artificial Intelegence, Mistral, Ollama

Abstrak: Website akademik berfungsi sebagai pusat layanan informasi institusi pendidikan tinggi, mencakup pengelolaan akademik, keuangan, dan komunikasi. Namun, sistem ini semakin rentan terhadap serangan siber akibat tingginya eksposur terhadap internet dan lemahnya proteksi terhadap celah keamanan. Penelitian ini mengusulkan solusi terintegrasi berupa pemindaian otomatis menggunakan OWASP ZAP, yang dikombinasikan dengan model kecerdasan buatan lokal Mistral melalui platform Ollama. Seluruh proses diotomatisasi menggunakan bahasa pemrograman Python, mencakup tahapan spidering, pemindaian aktif, ekstraksi hasil dalam format JSON, hingga pemberian rekomendasi mitigasi secara otomatis oleh AI. Objek penelitian adalah website Global Academic Information System, dengan hasil pengujian menunjukkan total 193 kerentanan, terdiri dari 4 risiko tinggi, 8 sedang, 111 rendah, dan 70 informasional. Setiap kerentanan diproses oleh AI untuk menghasilkan rekomendasi teknis spesifik, seperti penambahan header keamanan, token CSRF, serta pengaturan cookie yang aman. Seluruh hasil dikompilasi otomatis ke dalam laporan Excel yang siap digunakan oleh pengembang. Pendekatan ini terbukti efektif dalam meningkatkan efisiensi proses audit keamanan, mengurangi beban kerja manual, serta menjaga privasi data karena sistem berjalan secara lokal tanpa ketergantungan layanan cloud. Penelitian ini menunjukkan bahwa integrasi antara metode standar OWASP dengan AI lokal mampu menjadi solusi praktis, adaptif, dan mandiri dalam pengujian keamanan aplikasi web.

Kata kunci: OWASP ZAP, Keamanan Website, Kecerdasan Buatan Lokal, Mistral, Ollama

1. Pendahuluan

Kemajuan signifikan dalam bidang Teknologi Informasi dan Komunikasi (TIK) telah berdampak luas terhadap berbagai aspek kehidupan, termasuk sektor pendidikan dan keamanan digital [1]. Seiring dengan meluasnya adopsi teknologi, sistem informasi kini berperan sentral dalam mendukung aktivitas digital, mulai dari manajemen data hingga proses otomasi layanan [2]. Dalam konteks keamanan siber, pertumbuhan penggunaan aplikasi *web* memunculkan berbagai tantangan terhadap perlindungan data, yang salah satunya direspons melalui pendekatan sistematis seperti *Open Web Application Security Project* (OWASP), sebuah *framework* yang dikembangkan oleh organisasi nirlaba untuk meningkatkan keamanan perangkat lunak berbasis *web* [3]. Salah satu kontribusi populernya adalah publikasi OWASP Top 10, yaitu daftar kerentanan paling umum yang sering dijadikan acuan dalam pengujian keamanan aplikasi oleh para profesional keamanan siber [4, 5]. Di sisi lain, seiring meningkatnya ancaman serangan berbasis web seperti XSS dan SQL injection, pemanfaatan teknologi kecerdasan buatan (AI) menjadi pendekatan baru yang mampu meningkatkan efektivitas deteksi ancaman tersebut secara otomatis [6]. Model klasifikasi gambar yang dikembangkan menggunakan platform *Teachable Machine* terbukti mampu mendeteksi objek secara akurat dengan arsitektur ringan dan implementasi lokal [7], sehingga berpotensi besar untuk diterapkan dalam sistem keamanan siber yang membutuhkan proses analisis cepat dan tetap menjaga privasi data internal. Di sisi pengembangan sistem berbasis web, pendekatan sistem pendukung keputusan (SPK) juga telah dimanfaatkan dalam berbagai kebutuhan praktis. Penelitian oleh Ryando dkk. Pada tahun 2023 [8] merancang SPK pemilihan sepeda motor second terbaik menggunakan metode AHP dan TOPSIS yang diimplementasikan dengan bahasa pemrograman PHP dan database MySQL, guna mempermudah proses pengambilan keputusan secara objektif. Sementara itu, dalam pengembangan sistem informasi berbasis *web*, pendekatan *waterfall* dan penggunaan bahasa pemrograman PHP serta MySQL telah banyak digunakan untuk membangun platform digital adaptif, sebagaimana diterapkan dalam sistem penjualan online Toko Metro Snack yang meningkatkan efisiensi transaksi dan pengelolaan data [9]. Oleh karena itu, integrasi antara pendekatan standar seperti OWASP dengan kecanggihan *Artificial Intelligence* lokal menjadi strategi penting dalam menganalisis dan meningkatkan keamanan *website* secara menyeluruh.

Namun demikian, di balik manfaat besar yang ditawarkan oleh sistem informasi berbasis *web*, terdapat potensi ancaman serius berupa celah keamanan (*vulnerabilities*) yang kerap dimanfaatkan oleh pihak tidak bertanggung jawab. Serangan siber seperti *Cross-Site Scripting (XSS)*, SQL Injection, dan *Broken Access Control* menjadi bentuk ancaman yang umum ditemukan pada berbagai sistem aplikasi, termasuk di lingkungan pendidikan, pemerintahan, dan organisasi publik [10, 11]. Temuan dari berbagai penelitian menunjukkan bahwa banyak aplikasi web masih belum memiliki sistem perlindungan yang memadai karena kurangnya pengujian keamanan secara berkala [12]. Hal ini dapat menyebabkan kebocoran data sensitif, terganggunya layanan sistem, hingga menurunnya kepercayaan pengguna. Oleh sebab itu, diperlukan penguatan praktik *security assessment* dengan pendekatan terstruktur seperti OWASP, yang tidak hanya mengidentifikasi celah keamanan namun juga menghitung tingkat risiko untuk menentukan prioritas penanganan [13]. Penerapan *tools* seperti OWASP ZAP telah terbukti mampu mendeteksi berbagai kerentanan dalam sistem informasi akademik dan pemerintahan secara efektif, sehingga dapat memberikan rekomendasi perbaikan yang relevan [14]. Dengan integrasi metode standar OWASP dan pemanfaatan teknologi *Artificial Intelligence* lokal, maka pengujian keamanan website dapat dilakukan lebih komprehensif, adaptif, dan tetap menjaga kendali atas data secara penuh di lingkungan internal.

Beberapa studi terdahulu menunjukkan bahwa pendekatan OWASP telah banyak dimanfaatkan dalam proses identifikasi dan evaluasi kerentanan sistem berbasis *web*. Misalnya, penelitian oleh Riandhanu pada tahun 2022 [15] berhasil mendeteksi kelemahan pada aplikasi absensi *online* dengan menggabungkan metode OWASP Top 10 dan *tools Netsparke*r, serta menghasilkan rekomendasi perbaikan yang signifikan. Penelitian serupa oleh Zaini dan Wijanarko pada tahun 2023 [16] mengungkap celah seperti SQL Injection dan *Local File Inclusion* pada sistem Penerimaan Mahasiswa Baru, yang menunjukkan bahwa OWASP-ZAP efektif untuk menemukan celah tingkat menengah. Nurjannah dan Muni pada tahun 2024 [17] juga menerapkan OWASP-ZAP pada *website* sekolah dan berhasil mengidentifikasi beberapa kelemahan yang berpotensi dimanfaatkan oleh pihak tidak bertanggung jawab. Hal ini memperkuat pentingnya pengujian keamanan rutin sebagai bagian dari manajemen risiko TI. Selain pendekatan OWASP, Wibawa pada tahun 2023 [18] meneliti peran bahasa pemrograman Python dalam pengembangan *chatbot* otomatisasi administratif, menunjukkan bagaimana Python mendukung efisiensi tugas rutin yang dapat diadaptasi dalam pengelolaan sistem keamanan. Penelitian oleh Albert dan Voutama pada tahun 2025 [19] bahkan mengembangkan *chatbot* lokal berbasis PDF dengan pendekatan *Retrieval-Augmented Generation* (RAG), menggunakan *platform* Ollama, yang menegaskan pentingnya solusi privat tanpa ketergantungan pada *cloud*. Meskipun demikian, sebagian besar penelitian sebelumnya masih mengandalkan tahapan manual atau semi-otomatis dalam mendeteksi kerentanan. Belum banyak yang menggabungkan metode standar seperti OWASP dengan kecerdasan buatan lokal secara penuh dalam satu sistem otomatis. Inilah yang menjadi celah kontribusi dalam penelitian ini—yakni merancang proses pengujian yang dapat berjalan secara mandiri mulai dari scanning hingga penyusunan rekomendasi perbaikan berbasis *Artificial Intelligence* lokal.

Berbeda dari penelitian sebelumnya yang hanya mengandalkan pendekatan manual atau semi-otomatis dalam mendeteksi kerentanan menggunakan OWASP ZAP, penelitian ini mengintegrasikan sistem pengujian otomatis penuh yang tidak hanya mendeteksi kerentanan, tetapi juga menghasilkan rekomendasi mitigasi secara langsung menggunakan model *Artificial Intelligence* lokal. Ruang lingkup penelitian dibatasi pada pengujian keamanan terhadap website Global Academic Information System menggunakan standar OWASP Top 10, dengan proses otomatisasi mencakup pemindaian, ekstraksi data, hingga pemberian rekomendasi berbasis AI yang dijalankan secara lokal tanpa koneksi ke layanan *cloud*.

Metode yang digunakan dalam penelitian ini adalah pendekatan studi kasus dengan implementasi eksperimental. Penelitian dilakukan secara praktis terhadap satu objek sistem nyata, yaitu *website* Global Academic Information System, dengan tahapan yang mencakup pengumpulan informasi awal *(footprinting)*, pemindaian kerentanan menggunakan OWASP ZAP, serta analisis dan pemberian rekomendasi mitigasi otomatis melalui model kecerdasan buatan lokal Mistral. Semua tahapan diotomatisasi menggunakan skrip Python yang mengintegrasikan proses *scanning*, ekstraksi hasil, pemanggilan model AI, hingga penyusunan laporan. Pendekatan ini bertujuan untuk menghasilkan model evaluasi keamanan yang efisien, adaptif, dan privat.

Tujuan utama dari penelitian ini adalah untuk menganalisis tingkat keamanan website Global Academic Information System dengan mengidentifikasi berbagai kerentanan berbasis OWASP Top 10 dan memberikan rekomendasi mitigasi secara otomatis melalui model AI lokal. Selain itu, penelitian ini bertujuan merancang sebuah sistem terotomatisasi yang mampu mengintegrasikan proses pemindaian keamanan dan pemberian rekomendasi perbaikan dalam satu eksekusi, sehingga dapat membantu pengembang maupun auditor dalam mempercepat proses audit keamanan tanpa bergantung pada layanan *cloud public*, serta memastikan keamanan informasi sensitif tetap berada di lingkungan internal organisasi.

2. Bahan dan Metode

2.1 Objek Penelitian

Objek yang menjadi fokus dalam penelitian ini adalah *website* Global Academic Information System yang digunakan oleh civitas akademik Institut Teknologi dan Bisnis Bina Sarana Global. *Website* Global Academic Information System digunakan untuk aktivitas akademik seperti pengisian Kartu Rencana Studi (KRS), Kartu Hasil Studi (KHS), cek nilai, hingga administrasi keuangan mahasiswa. Karena tingginya sensitivitas data dan akses pengguna, website ini menjadi objek strategis dalam pengujian keamanan berbasis OWASP.

2.2 Metode Pengumpulan Informasi

Tahapan awal dalam penelitian ini adalah *intelligence gathering* dan *footprinting*, yang dilakukan untuk mendapatkan permukaan serangan (*attack surface*) dari *website* Global Academic Information System. Teknik yang digunakan mencakup pendekatan pasif dan aktif:

1. Teknik pasif dilakukan tanpa interaksi langsung dengan sistem target, seperti penggunaan dnsdumpster atau informasi publik dari DNS.
2. Teknik aktif dilakukan dengan melakukan interaksi langsung dengan server target untuk memperoleh data yang lebih spesifik.

Tools yang digunakan meliputi:

1. Nmap (aktif): digunakan untuk mendeteksi port aktif dan layanan yang berjalan di balik domain Global Academic Information System.
2. Subfinder (pasif): digunakan untuk enumerasi subdomain aktif yang terasosiasi dengan domain utama.
3. WhatWeb (aktif): digunakan untuk mendeteksi teknologi web seperti CMS, framework, dan web server.

Informasi yang diperoleh dari tahap ini digunakan sebagai dasar untuk mengarahkan proses pemindaian kerentanan berikutnya secara lebih terfokus dan efisien.

2.3 Tahap Pemindaian Kerentanan dan Rekomendasi Otomatis

Tahapan inti dalam penelitian ini dilakukan secara otomatis menggunakan skrip Python yang terhubung dengan OWASP ZAP dan Ollama. Alur kerjanya dijelaskan sebagai berikut:

1. *Spidering & Active Scan Otomatis*
   Python menginstruksikan OWASP ZAP untuk melakukan pemindaian menyeluruh ke struktur URL dari Global Academic Information System secara aktif dan pasif.
2. Ekstraksi Hasil JSON
   Semua hasil deteksi kerentanan diekstrak dalam format JSON dan dikelompokkan berdasarkan tingkat risiko serta jenis serangan (misalnya: XSS, Cookie insecure, Missing headers).
3. Integrasi dengan Artificial Intelligence Lokal Mistral
   Nama kerentanan dan deskripsi dikirim ke model Mistral melalui Ollama (dijalankan secara lokal). Untuk setiap entri kerentanan, Python menghasilkan prompt dalam bentuk deskriptif yang dikirim ke Mistral melalui API lokal Ollama. Contoh format prompt yang digunakan:
   a) Nama Kerentanan: Missing Anti-clickjacking Header
   b) Deskripsi: Sistem tidak memiliki header X-Frame-Options yang melindungi dari clickjacking.
   c) Tugas: Berikan rekomendasi teknis mitigasi untuk kerentanan ini.

4. Penyimpanan Otomatis
   Seluruh hasil (kerentanan, risiko, rekomendasi AI) diolah menggunakan Pandas dan disimpan dalam file Excel terstruktur, serta log file dalam format teks.

Prompt engineering ini dirancang agar model Mistral mampu memahami konteks masalah dan memberikan solusi teknis yang spesifik. Output dari Mistral berupa respon teks yang berisi saran teknis mitigasi.

1. Validasi Rekomendasi AI
   Untuk memastikan keakuratan rekomendasi yang diberikan oleh Mistral, setiap hasil dibandingkan dengan referensi dari dokumentasi OWASP dan sumber-sumber praktik keamanan standar industri. Validasi dilakukan secara manual oleh peneliti dengan pendekatan cross-check menggunakan database kerentanan seperti OWASP Cheat Sheet Series, Mozilla Secure Headers, dan konfigurasi server yang direkomendasikan oleh CIS Benchmarks.
2. Penyimpanan Otomatis
   Hasil akhir yang mencakup nama kerentanan, tingkat risiko, URL target, deskripsi, dan rekomendasi teknis dari AI disimpan ke dalam file Excel terstruktur menggunakan Pandas dan openpyxl. Selain itu, setiap prompt dan output dari Mistral juga disimpan dalam log file teks sebagai arsip proses analisis.

2.4 Alat dan Sistem Operasi

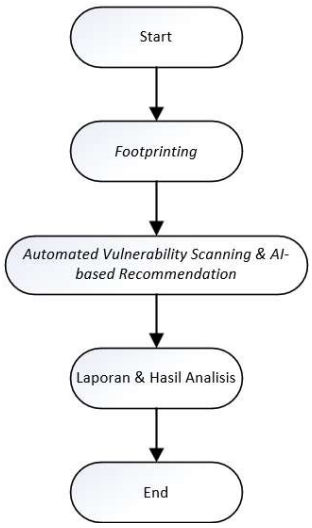
Pada pengujian kerentanan terhadap *website* Global Academic Information System, terdapat beberapa alat dan sistem operasi yang dibutuhkan untuk menganalisa kerentanan *website* yang terdapat pada Tabel 1.

Tabel 1. Alat dan spesifikasi sistem operasi

Nama perangkat dan pusaka	Versi	Deskripsi
Kali Linux	6.12.25	Sistem operasi yang digunakan untuk melakukan penetration testing
Nmap	7.95	Nmap digunakan sebagai explorasi network dan audit keamanan
Subfinder	2.7.1	Subfinder digunakan sebagai pengumpulan data suatu domain dan subdomain
Whatweb	0.5.5	Whatweb digunakan sebagai pendeteksi dan pengumpulan informasi dari teknologi web yang digunakan
VSCode	1.87	VSCode digunakan sebagai teks editor untuk membuat dan mengembangkan code untuk Tools yang dikembangkan
OWASP ZAP	2.16.1	OWASP ZAP digunakan sebagai penyedia API untuk untuk pemindaian kerentanan web
Ollama	0.6.8	Ollama digunakan sebagai platform untuk menjalankan model LLM (Large Language Model)
Microsoft Excel	16.0	Microsoft excel digunakan sebagai visualisasi dan dokumentasi akhir hasil audit
Python	3.12	Python digunakan sebagai bahasa pemrograman untuk mengembangkan Tools yang digunakan
Pandas	2.2.0	Pandas digunakan sebagai penyedia library Python untuk ekspor ke excel
Openpyxl	3.1.2	Openpyxl digunakan sebagai penyedia library pendukung Pandas untuk menyimpan file dalam format .xlsx
Mistral	7B	Mistral 7B digunakan untuk menghasilkan rekomendasi teknis secara otomatis

2.5 Alur Proses Sistem

Diagram alur pengujian keamanan website dalam penelitian ini digambarkan pada Gambar 1 berikut.



Gambar 1. Alur proses pengujian sistem

Deskripsi proses :

1. *Footprinting* dilakukan secara manual dan semi-otomatis menggunakan tools Kali Linux.
2. Pemindaian dan rekomendasi perbaikan dilakukan sekali eksekusi (one-click) menggunakan skrip Python yang menjalankan proses spidering, active scan, ekstraksi hasil, dan integrasi AI.
3. Sistem menyusun laporan akhir yang dapat langsung digunakan auditor atau pengembang untuk perbaikan keamanan.

3. Hasil

Untuk proses mengidentifikasi celah keamanan pada *website* Global Academic Information System, pengujian dilakukan melalui sejumlah tahapan yang telah dirancang secara sistematis berdasarkan metode yang telah dijelaskan pada bagian sebelumnya.

3.1. Footprinting

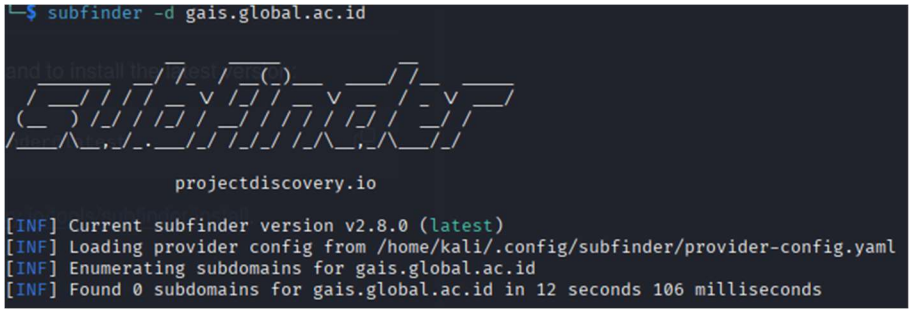
Tahap awal yang dilakukan adalah proses pengumpulan informasi dasar terhadap target sistem, yang dikenal dengan istilah *footprinting*. Tahapan ini bertujuan untuk memetakan permukaan serangan (*attack surface*) dan mengidentifikasi aset-aset yang terbuka serta dapat diakses publik. Pada penelitian ini, digunakan tiga *tools* utama untuk proses *footprinting*, yaitu Nmap, Whatweb, dan Subfinder, yang ketiganya dijalankan melalui sistem operasi Kali Linux. Berikut adalah hasil informasi dari setiap *tools footprinting* yang digunakan:

1. Nmap
   Dapat dilihat pada Tabel 2, proses ini dilakukan menggunakan *tool* Nmap dengan perintah nmap -sC -sV gais.global.ac.id melalui terminal di sistem operasi Kali Linux. Hasil pemindaian menunjukkan beberapa port terbuka seperti 22/tcp (SSH), 80/tcp dan 443/tcp (HTTP/HTTPS), serta 3306/tcp (MySQL), lengkap dengan informasi versi layanan yang berjalan. Data ini digunakan untuk mengidentifikasi potensi permukaan serangan pada sistem target.

Tabel 2. Hasil pengumpulan informasi menggunakan *tools* Nmap

Port	State	Service	Version
22/tcp	open	ssh	OpenSSH 7.4
80/tcp	open	http	Apache 2.4.62/PHP 7.4.33
443/tcp	open	https/ssl	Apache 2.4.62/PHP 7.4.33
3306/tcp	open	mysql	MariaDB 5.5.68

2. Subfinder



Gambar 2. Hasil pencarian sub domain menggunakan *tools* subfinder

Dapat dilihat pada Gambar 2 tidak ditemukan adanya sub domain yang terkait dengan *website* gais.global.ac.id, hasil tersebut menunjukkan bahwa peluang bagi penyerang untuk mengeksploitasi sistem-sistem tersebut juga lebih kecil.

3. WhatWeb

Gambar 3. Identifikasi CMS menggunakan *tools* whatweb

Hasil identifikasi teknologi menggunakan WhatWeb pada Gambar 3 menunjukkan bahwa website *gais.global.ac.id* menggunakan *framework Bootstrap* dan *CodeIgniter* di sisi aplikasi, serta PHP versi 7.4.33 sebagai bahasa *server-side*. Server web yang digunakan adalah Apache 2.4.62 dengan dukungan OpenSSL 1.0.2. Website ini juga memanfaatkan HTML5 dan pustaka *JavaScript jQuery*. Informasi ini memberikan gambaran awal mengenai teknologi yang digunakan, yang dapat menjadi dasar untuk mengidentifikasi potensi kerentanan pada komponen-komponen tersebut.

Dari hasil pengujian yang ditunjukkan pada Tabel 2, Gambar 2, dan Gambar 3, proses *footprinting* terhadap *website* Global Academic Information System berhasil mengidentifikasi sejumlah informasi sensitif, seperti *resource* yang digunakan untuk membuat *website,* alamat IP server, nama *domain*, versi sistem operasi, *port* terbuka, layanan yang berjalan beserta versinya, serta lokasi *hosting*. Informasi ini diperoleh melalui tools Nmap, Whatweb, dan Subfinder. Ditemukannya data teknis secara terbuka ini menunjukkan bahwa sistem belum sepenuhnya terlindungi dari potensi *reconnaissance* oleh pihak tidak bertanggung jawab. Maka perlu diterapkan pengamanan seperti WHOIS *protection*, penutupan *port* yang tidak diperlukan, penyembunyian versi layanan, dan konfigurasi server yang lebih aman untuk mencegah eksploitasi lanjutan.

3.2. Hasil Pemindaian Kerentanan dan Rekomendasi AI

Setelah proses *footprinting* selesai, dilakukan pemindaian kerentanan menggunakan skrip Python yang terintegrasi dengan OWASP ZAP dan model *Artificial Intelligence* lokal Mistral melalui *platform* Ollama. Proses ini dilakukan dalam satu kali eksekusi otomatis. Adapun langkah – langkah yang dilakukan selama skrip Python berjalan sebagai berikut:

```
\Tools> python main.py
```

```python
# Konfigurasi OWASP ZAP dan target
ZAP_API_KEY = 'e5i615df82d8c7m6r9cojs3bdu'
ZAP_ADDRESS = 'localhost'
ZAP_PORT = '8080'
TARGET = 'https://gais.global.ac.id'

# Konfigurasi Model LLM Mistral Untuk Mencari Rekomendasi Perbaikan
OLLAMA_URL = "http://localhost:11434/api/generate"
MODEL_NAME = "mistral"

zap = ZAPv2(apikey=ZAP_API_KEY, proxies={'http': f'http://{ZAP_ADDRESS}:{ZAP_PORT}',
                                         'https': f'http://{ZAP_ADDRESS}:{ZAP_PORT}'})
```

Gambar 4. Perintah Phyton untuk menjalankan skrip

1. Sistem dijalankan melalui perintah *python main.py*, yang otomatis menginisialisasi integrasi antara Python, OWASP ZAP *API*, dan model *Artificial Intelligence* lokal Mistral melalui Ollama. Perintah dapat dilihat pada Gambar 4.

```
ok\Tools> python main.py
[+] Spidering target: https://gais.global.ac.id
[+] Spidering selesai.
[+] Active scanning target: https://gais.global.ac.id
    Scan progress: 2%
    Scan progress: 4%
    Scan progress: 7%
    Scan progress: 19%
    Scan progress: 27%
    Scan progress: 29%
    Scan progress: 37%
    Scan progress: 42%
    Scan progress: 50%
    Scan progress: 50%
    Scan progress: 61%
    Scan progress: 67%
    Scan progress: 77%
    Scan progress: 93%
```

```python
# === Melakukan Crawling Untuk Mencari struktur URL dan input-input form ===
def start_spider(target):
    print(f"[+] Spidering target: {target}")
    scan_id = zap.spider.scan(target)
    time.sleep(2)
    while int(zap.spider.status(scan_id)) < 100:
        print(f"    Spider progress: {zap.spider.status(scan_id)}%")
        time.sleep(2)
    print("[+] Spidering selesai.")
```

Gambar 5. Sistem melakukan proses *crawling* dan *active scan*

2. OWASP ZAP memulai tahap *spidering*, yaitu proses penelusuran struktur URL *website* target (https://gais.global.ac.id) pada Gambar 5, untuk mengidentifikasi halaman dan input form yang relevan.

3. Setelah *spidering* selesai, dilanjutkan dengan *active scan* untuk mengidentifikasi kerentanan yang terdapat pada halaman-halaman yang terdeteksi.

```
[+] Spidering target: https://gais.global.ac.id
[+] Spidering selesai.
[+] Active scanning target: https://gais.global.ac.id
    Scan progress: 2%
    Scan progress: 4%
    Scan progress: 7%
    Scan progress: 19%
    Scan progress: 27%
    Scan progress: 29%
    Scan progress: 37%
    Scan progress: 42%
    Scan progress: 50%
    Scan progress: 50%
    Scan progress: 61%
    Scan progress: 67%
    Scan progress: 77%
    Scan progress: 93%
[+] Active scan selesai.
[+] Mengambil hasil scan...
[+] Ditemukan 193 kerentanan.
```

```python
# === Dilanjutkan Dengan Active Scan Untuk Menguji Berbagai Kerentanan ===
def start_active_scan(target):
    print(f"[+] Active scanning target: {target}")
    scan_id = zap.ascan.scan(target)
    time.sleep(2)
    while int(zap.ascan.status(scan_id)) < 100:
        print(f"    Scan progress: {zap.ascan.status(scan_id)}%")
        time.sleep(5)
    print("[+] Active scan selesai.")

# === Mengambil Hasil Kerentanan Dari ZAP Setelah Pemindaian Selesai ===
def get_alerts():
    print("[+] Mengambil hasil scan...")
    alerts = zap.core.alerts(baseurl=TARGET)
    print(f"[+] Ditemukan {len(alerts)} kerentanan.")
    return alerts
```

Gambar 6. Sistem berjalan secara otomatis

```
{
  "sourceid": "3",
  "other": "\ncookie:ci_session",
  "method": "GET",
  "evidence": "9810bf5ja5mr9jtfvi1e3lucmclb6i0a",
  "pluginId": "10112",
  "cweid": "-1",
  "confidence": "Medium",
  "sourceMessageId": 7,
  "wascid": "-1",
  "description": "The given response has been identified as containing a session management token. The 'Other Info' field contains a set of heade
  "messageId": "7",
  "inputVector": "",
  "url": "https://gais.global.ac.id",
  "tags": {},
  "reference": "https://www.zaproxy.org/docs/desktop/addons/authentication-helper/session-mgmt-id",
  "solution": "This is an informational alert rather than a vulnerability and so there is nothing to fix.",
  "alert": "Session Management Response Identified",
  "param": "ci_session",
  "attack": "",
  "name": "Session Management Response Identified",
  "risk": "Informational",
  "id": "3",
  "alertRef": "10112"
},
```

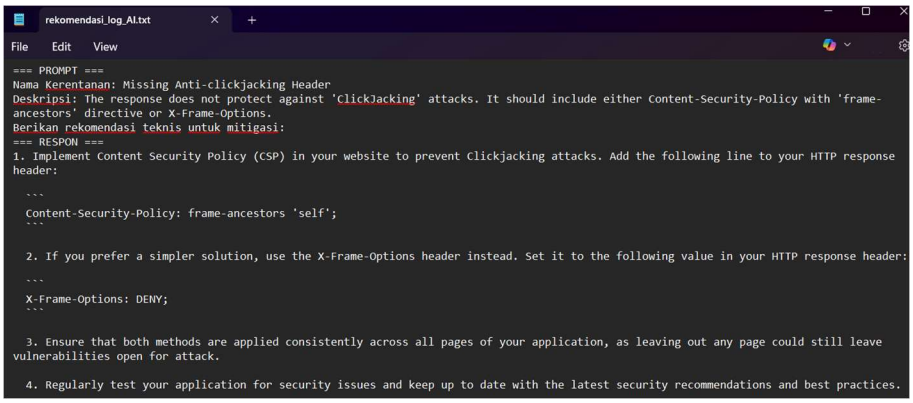

Gambar 7. Salah satu kerentanan dari hasil *scanning* dalam format JSON

4. Setelah sistem melakukan *active scan* selesai, sistem secara otomatis mengambil seluruh hasil kerentanan yang terdeteksi oleh ZAP dalam format JSON, yang mencakup informasi nama kerentanan, tingkat risiko, deskripsi, dan URL terkait. Dapat dilihat pada Gambar 6, sistem berhasil menemukan total 193 kerentanan. Salah satu contoh hasil deteksi kerentanan dalam format JSON ditampilkan pada Gambar 7 untuk memperlihatkan struktur data yang dihasilkan oleh proses pemindaian.

```
[+] Ditemukan 193 kerentanan.
[+] Mulai Melakukan Report Hasil Scan dan Rekomendasi Perbaikan
[+] Memproses: Authentication Request Identified
    Rekomendasi: 1. Implement Two-Factor Authentication (2FA): Add an additional layer ...
[+] Mulai Melakukan Report Hasil Scan dan Rekomendasi Perbaikan
[+] Memproses: Content Security Policy (CSP) Header Not Set
    Rekomendasi: 1. Implement Content Security Policy (CSP) in your web application by ...
```

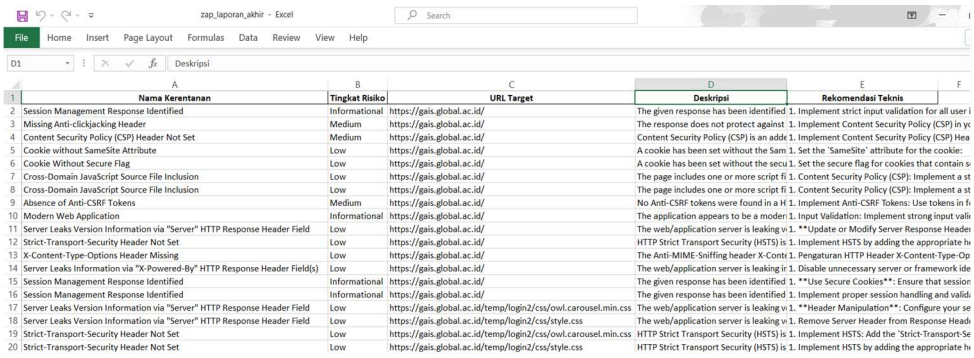
```python
# === Mengambil Log Cara Kerja Hasil Scan Di Kirim Ke Ollama ===
def log_to_file(prompt, response, filename="rekomendasi_log.txt"):
    with open(filename, "a", encoding="utf-8") as f:
        f.write("=== PROMPT ===\n")
        f.write(prompt + "\n")
        f.write("=== RESPON ===\n")
        f.write(response + "\n\n")

# === Mengirim Nama dan Deskripsi Kerentanan Kepada Ollama Untuk Rekomendasi Perbaikan ===
def generate_local_recommendation(alert_name, description):
    prompt = (
        f"Nama Kerentanan: {alert_name}\n"
        f"Deskripsi: {description}\n"
        f"Berikan rekomendasi teknis untuk mitigasi:"
    )
    try:
        response = requests.post(OLLAMA_URL, json={
            "model": MODEL_NAME,
            "prompt": prompt,
            "stream": False
        })
        if response.status_code == 200:
            hasil = response.json().get("response", "").strip()
            log_to_file(prompt, hasil)  # Log ke file setelah dapat jawaban
            return hasil
        else:
            log_to_file(prompt, "Gagal mengambil rekomendasi dari model lokal.")
            return "Gagal mengambil rekomendasi dari model lokal."
    except Exception as e:
        log_to_file(prompt, f"Terjadi kesalahan: {e}")
        return f"Terjadi kesalahan saat mengambil rekomendasi: {e}"
```

Gambar 8. Proses input dalam bentuk *prompt* ke *Artificial Intelligence* local



Gambar 9. Log respon dari *prompt* ke *Artificial Intelligence* lokal

5. Data JSON tersebut dibaca dan diubah menjadi *prompt* yang dikirim ke model Mistral melalui *API* Ollama, dilanjutkan memproses input dan memberikan respon berupa rekomendasi teknis mitigasi untuk tiap kerentanan. Proses dapat dilihat pada Gambar 8 dan hasil dari *prompt* dapat dilihat pada Gambar 9.

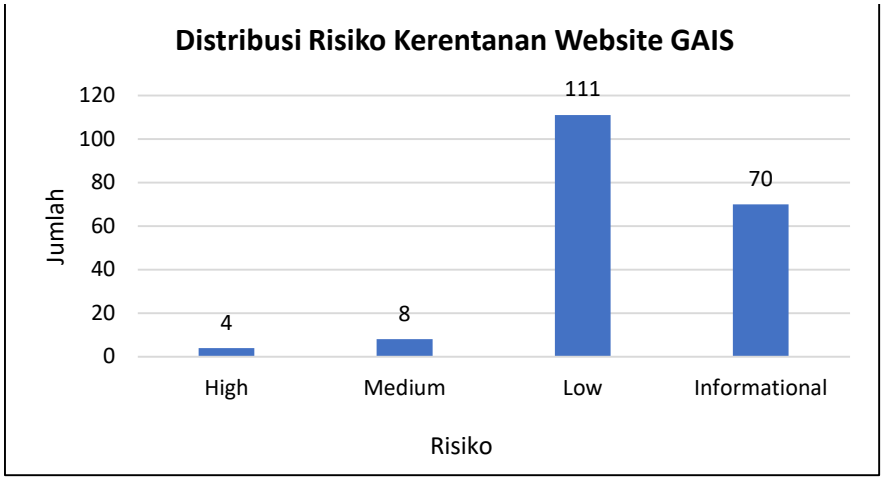


	Nama Kerentanan	Tingkat Risiko	URL Target	Deskripsi	Rekomendasi Teknis
2	Session Management Response Identified	Informational	https://gais.global.ac.id/	The given response has been identified	1. Implement strict input validation for all user i
3	Missing Anti-clickjacking Header	Medium	https://gais.global.ac.id/	The response does not protect against	1. Implement Content Security Policy (CSP) in yo
4	Content Security Policy (CSP) Header Not Set	Medium	https://gais.global.ac.id/	Content Security Policy (CSP) is an adde	1. Implement Content Security Policy (CSP) Hea
5	Cookie without SameSite Attribute	Low	https://gais.global.ac.id/	A cookie has been set without the Sam	1. Set the `SameSite` attribute for the cookie:
6	Cookie Without Secure Flag	Low	https://gais.global.ac.id/	A cookie has been set without the secu	1. Set the secure flag for cookies that contain s
7	Cross-Domain JavaScript Source File Inclusion	Low	https://gais.global.ac.id/	The page includes one or more script fi	1. Content Security Policy (CSP): Implement a st
8	Cross-Domain JavaScript Source File Inclusion	Low	https://gais.global.ac.id/	The page includes one or more script fi	1. Content Security Policy (CSP): Implement a st
9	Absence of Anti-CSRF Tokens	Medium	https://gais.global.ac.id/	No Anti-CSRF tokens were found in a H	1. Implement Anti-CSRF Tokens: Use tokens in fo
10	Modern Web Application	Informational	https://gais.global.ac.id/	The application appears to be a moder	1. Input Validation: Implement strong input valid
11	Server Leaks Version Information via "Server" HTTP Response Header Field	Low	https://gais.global.ac.id/	The web/application server is leaking ve	1. **Update or Modify Server Response Header
12	Strict-Transport-Security Header Not Set	Low	https://gais.global.ac.id/	HTTP Strict Transport Security (HSTS) is	1. Implement HSTS by adding the appropriate he
13	X-Content-Type-Options Header Missing	Low	https://gais.global.ac.id/	The Anti-MIME-Sniffing header X-Conte	1. Pengaturan HTTP Header X-Content-Type-Op
14	Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low	https://gais.global.ac.id/	The web/application server is leaking in	1. Disable unnecessary server or framework ide
15	Session Management Response Identified	Informational	https://gais.global.ac.id/	The given response has been identified	1. **Use Secure Cookies**: Ensure that session
16	Session Management Response Identified	Informational	https://gais.global.ac.id/	The given response has been identified	1. Implement proper session handling and valida
17	Server Leaks Version Information via "Server" HTTP Response Header Field	Low	https://gais.global.ac.id/temp/login2/css/owl.carousel.min.css	The web/application server is leaking ve	1. **Header Manipulation**: Configure your se
18	Server Leaks Version Information via "Server" HTTP Response Header Field	Low	https://gais.global.ac.id/temp/login2/css/style.css	The web/application server is leaking ve	1. Remove Server Header from Response Head
19	Strict-Transport-Security Header Not Set	Low	https://gais.global.ac.id/temp/login2/css/owl.carousel.min.css	HTTP Strict Transport Security (HSTS) is	1. Implement HSTS: Add the `Strict-Transport-Se
20	Strict-Transport-Security Header Not Set	Low	https://gais.global.ac.id/temp/login2/css/style.css	HTTP Strict Transport Security (HSTS) is	1. Implement HSTS by adding the appropriate he

Gambar 10. Hasil laporan kerentanan

6. Seluruh hasil kerentanan dan rekomendasi disusun secara otomatis ke dalam file Excel (*zap_laporan_akhir.xlsx*) pada Gambar 10.

Setelah seluruh tahapan otomatisasi dijalankan—mulai dari proses *spidering*, *active scan*, pengambilan hasil dalam format JSON, hingga pemberian rekomendasi berbasis AI lokal—sistem berhasil mengidentifikasi total 193 kerentanan dari website target. Kerentanan-kerentanan ini mencakup berbagai aspek keamanan aplikasi web, seperti konfigurasi header yang tidak aman, ketiadaan token CSRF, serta penggunaan *cookie* yang rentan.



Gambar 11. Diagram distribusi risiko kerentanan website GAIS

Setiap kerentanan yang ditemukan kemudian dianalisis dan diklasifikasikan tingkat risikonya menjadi empat kategori, yaitu tinggi *(high)*, sedang *(medium)*, rendah *(low)*, dan informasional *(informational)*. Penentuan kategori risiko ini mengacu pada OWASP *Risk Rating Methodology* [20], yang menilai kerentanan berdasarkan kombinasi faktor kemungkinan eksploitasi dan dampak terhadap aspek kerahasiaan, integritas, serta ketersediaan sistem. Visualisasi distribusi dari klasifikasi risiko ini dapat dilihat pada Gambar 11, dengan rincian: 4 risiko tinggi, 8 risiko sedang, 111 risiko rendah, dan 70 bersifat informasional.

Beberapa hasil dari OWASP ZAP secara *default* dikategorikan sebagai risiko sedang (*medium*), namun dilakukan penyesuaian berdasarkan dokumentasi resmi OWASP Top 10. Misalnya, kerentanan *Absence of Anti-CSRF Tokens* dinaikkan menjadi risiko tinggi (*high*) karena berpotensi menyebabkan serangan terhadap validitas sesi pengguna. Penyesuaian ini bertujuan agar klasifikasi risiko mencerminkan tingkat urgensi penanganan yang lebih tepat dalam konteks pengujian.

Seluruh kerentanan yang telah diklasifikasikan kemudian dikirim ke model *Artificial Intelligence* lokal Mistral untuk mendapatkan rekomendasi mitigasi teknis yang relevan. Adapun hasil analisis ditampilkan pada Tabel 3 dan telah dipetakan berdasarkan kategori OWASP Top 10.

Tabel 3. Pemetaan Kerentanan *website* Global Academic Information System terhadap OWASP Top 10

Nama Kerentanan	Risiko	URL Terdampak	Kategori OWASP Top 10	Rekomendasi Teknis Artificial Intelligence Lokal
Cookie Without Secure Flag	Medium	https://gais.global.ac.id/login	A05: Security Misconfiguration	Tambahkan atribut Secure dan HttpOnly pada Set-Cookie header
Missing Anti-clickjacking Header	Low	https://gais.global.ac.id/index	A05: Security Misconfiguration	Tambahkan header X-Frame-Options: DENY untuk mencegah iframe
Server Leaks Version Info via HTTP Header	Low	https://gais.global.ac.id/	A06: Vulnerable and Outdated Components	Sembunyikan atau ubah nilai header Server untuk menghindari fingerprinting
Absence of Anti-CSRF Tokens	High	https://gais.global.ac.id/forms/krs	A01: Broken Access Control	Tambahkan token CSRF unik pada setiap form dan validasi di sisi server, agar penyerang tidak dapat merekayasa sebuah halaman web eksternal yang memodifikasi data tanpa persetujuan atau pengetahuan pengguna
X-Content-Type-Options Header Missing	Medium	https://gais.global.ac.id/assets/bootstrap.js	A05: Security Misconfiguration	Tambahkan header X-Content-Type-Options: nosniff pada respons HTTP
Authentication Request Identified	Informational	https://gais.global.ac.id/login/validasi	A06: Vulnerable and Outdated Components	Gunakan otentikasi yang kuat dan tidak terbuka untuk publik
Session Management Response Identified	Informational	https://gais.global.ac.id/	A02: Cryptographic Failures	Implement strict input validation untuk seluruh parameter session
Cross-Domain JavaScript Source File Inclusion	Low	https://gais.global.ac.id/	A06: Vulnerable and Outdated Components	Gunakan Content Security Policy (CSP) untuk membatasi sumber skrip eksternal
Cookie without SameSite Attribute	Low	https://gais.global.ac.id/	A05: Security Misconfiguration	Set atribut SameSite pada cookie untuk mencegah CSRF
Timestamp Disclosure Unix	Low	https://gais.global.ac.id/temp/assets/log_main.log	A06: Vulnerable and Outdated Components	Sembunyikan informasi timestamp dari file log yang dapat diakses publik

Melalui pendekatan otomatis ini, proses deteksi dan rekomendasi mitigasi dapat dilakukan secara efisien dalam satu kali eksekusi, sehingga sangat bermanfaat bagi pengembang maupun auditor sistem.

4. Pembahasan

Berdasarkan hasil pemindaian, sebagian besar celah keamanan yang ditemukan berkaitan dengan kesalahan konfigurasi sistem (A05) dan lemahnya kontrol akses (A01) menurut klasifikasi OWASP Top 10. Meskipun platform Global Academic Information System telah menerapkan *HTTPS* dan autentikasi pengguna, hasil analisis menunjukkan bahwa aspek seperti konfigurasi *header* keamanan serta mekanisme validasi *input* belum sepenuhnya diterapkan dengan baik. Hal ini menunjukkan bahwa keberadaan fitur dasar keamanan tidak menjamin sistem terbebas dari kerentanan, terutama jika pengaturannya tidak dilakukan secara menyeluruh.

Keunggulan utama dari pendekatan yang diusulkan dalam penelitian ini terletak pada pemanfaatan model *Artificial Intelligence* lokal (Mistral) untuk memberikan rekomendasi teknis secara langsung terhadap setiap kerentanan yang ditemukan. Model grafAI ini digunakan dengan mengubah hasil pemindaian OWASP ZAP dalam format JSON menjadi input berupa *prompt* teks yang mendeskripsikan jenis kerentanan, nama, dan dampaknya. *Prompt* tersebut kemudian dikirim ke model Mistral melalui API Ollama secara lokal tanpa koneksi internet. Model akan merespons dengan rekomendasi mitigasi yang spesifik, teknis, dan sesuai konteks. Sebagai contoh, untuk kasus *CSRF*, AI menyarankan implementasi token acak dan validasi sisi server. Untuk kerentanan konfigurasi *header*, AI merekomendasikan pengaturan ulang nilai *header* seperti *X-Content-Type-Options* dan *Strict-Transport-Security* untuk memperkuat proteksi. Hasil rekomendasi ini selanjutnya disusun secara otomatis ke dalam laporan yang siap digunakan oleh pengembang sebagai acuan perbaikan.

Namun demikian, penggunaan AI lokal seperti Mistral juga memiliki trade-off. Keunggulannya terletak pada independensi dari layanan cloud sehingga menjamin privasi data dan keterulangan hasil secara konsisten. Akan tetapi, karena model dijalankan secara lokal, performa sistem sangat bergantung pada spesifikasi perangkat keras. Dalam penelitian ini, inferensi model berjalan dalam waktu relatif cepat karena ukuran model yang efisien (7B parameter), namun tetap membutuhkan alokasi memori di atas 6 GB serta pemrosesan CPU/GPU yang stabil selama proses pengolahan data.

Terkait efektivitas hasil rekomendasi AI, penelitian ini melakukan validasi manual terhadap beberapa contoh hasil dengan membandingkan rekomendasi Mistral terhadap praktik standar OWASP dan CIS Benchmark. Secara umum, rekomendasi yang dihasilkan AI cukup sesuai dengan prinsip-prinsip mitigasi yang direkomendasikan oleh komunitas keamanan, meskipun belum semuanya menyertakan contoh implementasi kode yang spesifik. Untuk itu, sistem ini lebih cocok digunakan sebagai *decision support tool* dan bukan sebagai pengganti penuh evaluasi oleh manusia.

Pendekatan yang diusulkan terbukti lebih unggul dibandingkan metode manual seperti yang digunakan dalam studi terdahulu, misalnya oleh Sabrina et al. [21], yang hanya menghasilkan laporan deteksi tanpa rekomendasi konkret. Dengan adanya integrasi OWASP ZAP dan *Artificial Intelligence* lokal, sistem ini mampu menghasilkan laporan yang tidak hanya informatif tetapi juga praktis dalam mendukung proses perbaikan keamanan aplikasi.

5. Kesimpulan dan Saran

5.1. Kesimpulan

Penelitian ini berhasil mengembangkan dan menerapkan sistem pengujian keamanan website yang bersifat otomatis dengan menggabungkan OWASP ZAP dan model

kecerdasan buatan lokal Mistral. Integrasi tersebut memungkinkan proses *spidering*, pemindaian aktif, ekstraksi hasil, serta penyusunan rekomendasi mitigasi dilakukan secara terstruktur dan efisien hanya melalui satu kali eksekusi. Hasil evaluasi pada *website* Global Academic Information System menunjukkan adanya 193 kerentanan, di mana sebagian besar termasuk dalam kategori kesalahan konfigurasi dan kontrol akses. Pendekatan lokal ini juga menjamin privasi data dan memungkinkan implementasi pada lingkungan sistem yang tidak terhubung ke *cloud*.

5.2. Saran

Berdasarkan hasil penelitian dan pengalaman implementasi sistem, penulis mengajukan beberapa saran untuk pengembangan lebih lanjut dan penerapan sistem dalam skala yang lebih luas:

1. Penerapan AI Lokal Berbasis Multi-Model (Ensemble)
   Penelitian selanjutnya dapat mengeksplorasi pendekatan ensemble dengan menggabungkan beberapa model lokal (seperti Mistral, LLaMA, dan Phi-2) untuk menghasilkan rekomendasi mitigasi yang lebih konsisten dan akurat. Dengan sistem voting atau confidence scoring, output rekomendasi dapat dipilih dari model yang paling relevan terhadap jenis kerentanan tertentu.
2. Pengujian Efektivitas Mitigasi Secara Langsung (Re-Scan Evaluation)
   Dalam pengembangan selanjutnya, sistem dapat dilengkapi dengan fitur re-scan otomatis untuk menguji apakah rekomendasi yang telah diterapkan benar-benar menutup celah. Dengan kata lain, AI tidak hanya menyarankan, tetapi juga mengevaluasi efektivitas mitigasinya secara langsung.
3. Integrasi Natural Language Explanation untuk User Non-Teknis
   Sistem dapat dikembangkan agar mampu mengubah hasil rekomendasi teknis menjadi versi narasi yang dapat dipahami oleh pihak non-teknis (misalnya manajemen atau auditor). Ini dapat dicapai dengan membangun modul parafrase AI yang menjelaskan risiko dan langkah mitigasi dengan bahasa alami dan contoh kasus.
4. Pengembangan Antarmuka Visual dengan Integrasi Log Real-time
   Disarankan untuk membangun antarmuka berbasis web menggunakan framework seperti Flask atau Django yang dapat menampilkan hasil pemindaian secara interaktif, termasuk status spidering, progres scanning, grafik distribusi risiko, serta rekomendasi AI. Visualisasi real-time ini dapat membantu tim pengembang dalam melakukan analisis kerentanan secara dinamis tanpa membuka file output secara manual.

Referensi

[1] E. Rohyadi and C. Atikah, "Peran Penting Teknologi Informasi dan Komunikasi (TIK) Dalam Pendidikan," *Pendas : J. Ilm. Pendidik. Dasar*, vol. 09, no. 04, pp. 752–764, 2024, https://journal.unpas.ac.id/index.php/pendas/article/view/18942.

[2] N. S. Nasabiyah *et al.*, "Peran Teknologi dan Komunikasi (TIK) dalam Proses Pembelajaran di MA Miftahul Ulum Kedungpanji," *J. Dewantara*, vol. 3, no. 3, pp. 195–208, 2022, https://doi.org/10.30640/dewantara.v3i3.2720.

[3] A. W. Kuncoro and F. Rahma, "Analisis Metode Open Web Application Security Project (OWASP) pada Pengujian Keamanan Website: Literature Review," *Automata*, vol. 3, no. 1, pp. 1–5, 2021, https://journal.uii.ac.id/AUTOMATA/article/view/21893

[4] N. Herawati, V. Budiyanto, and Umiringsih, "Analisis Keamanan Sebuah Domain Menggunakan Open Web Application Security Project (OWASP) Zap," *J. Teknol. Technoscientia*, vol. 15, no. 2, pp. 27–36, 2023, https://doi.org/10.34151/technoscientia.v15i2.4013.

[5] A. F. Hasibuan and D. Handoko, "Analisis Keretanan Website Dengan Aplikasi Owasp Zap," *J. Ilmu Komput. dan Sist. Inf.*, vol. 2, no. 2, pp. 257–270, 2023, https://doi.org/10.70340/jirsi.v2i2.51.

[6] R. Bakır, "UniEmbed: A Novel Approach to Detect XSS and SQL Injection Attacks Leveraging Multiple Feature Fusion with Machine Learning Techniques," *Arab. J. Sci. Eng.*, 2025. https://doi.org/10.1007/s13369-024-09916-4.

[7] I. Hidayatullah *et al.*, "Analisis Performa Deteksi Penyakit Padi Dengan Model Klasifikasi Gambar Menggunakan Teachable Machine," *Acad. J. Comput. Sci. Res.*, vol. 7, no. 1, pp. 1–6, 2025. http://dx.doi.org/10.38101/ajcsr.v7i1.15669

[8] M. B. Ryando, A. R. Mariana, and R. A. Hakim, "Sistem Pendukung Keputusan Pemilihan Sepeda Motor Second Terbaik di Kelas Matic 150cc Menggunakan Metode AHP dan TOPSIS," *Acad. J. Comput. Sci. Res.*, vol. 5, no. 1, p. 47, 2023. https://doi.org/10.38101/ajcsr.v5i1.611.

[9] N. T. Sunggono, D. Sofia, and A. Latif, "Penjualan Sembako Berbasis Web pada Toko Metro Snack," *J. Tren Bisnis Glob.*, vol. 2, no. 2, p. 42, 2022. https://doi.org/10.38101/jtbg.v2i2.574.

[10] A. Elanda and R. L. Buana, "Analisis Keamanan Sistem Informasi Berbasis Website Dengan Metode Open Web Application Security Project (OWASP) Versi 4: Systematic Review," *CESS (Journal Comput. Eng. Syst. Sci.*, vol. 5, no. 2, p. 185, 2020. https://doi.org/10.24114/cess.v5i2.17149.

[11] A. Dharmawan, Y. Prihati, and H. Listijo, "Penetration testing menggunakan OWASP top 10 pada domain xyz.ac.id," *Jelc*, vol. 8, no. 1, pp. 1–9, 2022. https://poltekstpaul.ac.id/jurnal/index.php/jelekn/article/view/455.

[12] S. Sabariman, H. Haeruddin, and D. Lee, "Analisis Kerentanan Aplikasi Akademik Berbasis Website Xyz Menggunakan Owasp," *J. Khatulistiwa Inform.*, vol. 11, no. 2, pp. 92–102, 2024. https://doi.org/10.31294/jki.v11i2.20194.

[13] D. Aryanti, Nurholis, and J. N. Utamajaya, "Analisis Kerentanan Keamanan Website Menggunakan Metode Owasp (Open Web Application Security Project) Pada Dinas Tenaga Kerja," *Syntax Fusion J. Nas. Indones.*, vol. 1, no. 3, pp. 238–248, 2021. https://doi.org/10.54543/fusion.v1i03.53.

[14] F. Nisa, N. S. Nurfebruary, and M. Ikhwani, "Analisis Keamanan Sistem Informasi Website Portal Akademik Universitas Malikussaleh Menggunakan OWASP ZAP," *J. Nas. Komputasi dan Teknol. Inf.*, vol. 7, no. 6, pp. 2003–2013, 2024. https://doi.org/10.32672/jnkti.v7i6.8345.

[15] I. O. Riandhanu, "Analisis Metode Open Web Application Security Project (OWASP) Menggunakan Penetration Testing pada Keamanan Website Absensi," *J. Inf. dan Teknol.*, vol. 4, no. 3, pp. 160–165, 2022. https://doi.org/10.37034/jidt.v4i3.236.

[16] A. Zaini and R. Wijanarko, "Analisis Keamanan Website Menggunakan Standar Keamanan Open Web Application Security Project (OWASP) Studi Kasus Website Penerimaan Mahasiswa Baru Universitas Wahid Hasyim Semarang," vol. 5, no. 2, 2023.

[17] Nurjannah and Abdul Muni, "Analisis Keamanan Website Sekolah Sman 1 Tempuling Dengan Menggunakan Open Web Application Security Project (Owasp)," *J. Perangkat Lunak*, vol. 6, no. 2, pp. 351–361, 2024. https://doi.org/10.32520/jupel.v6i2.3442.

[18] S. Wibawa, "Analisis Chatbot Otomatisasi Tugas Administratif dan Manajemen Dalam Lingkungan Digital Dengan Menggunakan Python," *Insantek*, vol. 4, no. 1, pp. 25–31, 2023. https://doi.org/10.31294/insantek.v4i1.2190.

[19] G. D. Albert and A. Voutama, "PENGEMBANGAN CHATBOT BERBASIS PDF MENGGUNAKAN LOCAL RETRIEVAL-AUGMENTED GENERATION (RAG) DAN OLLAMA," *J. Inform. dan Tek. Elektro Terap.*, vol. 13, no. 2, Apr. 2025. https://doi.org/10.23960/jitet.v13i2.6361.

[20] OWASP Foundation, "OWASP Risk Rating Methodology." Accessed: Jun. 22, 2025. [Online]. Available: https://owasp.org/www-community/OWASP_Risk_Rating_Methodology

[21] S. A. Febriani, A. Muni, B. Rianto, M. Jalil, and Chrismondari, "Analisis Kerentanan Keamanan Sistem Informasi Akademik Menggunakan Owasp-Zap Di Universitas Islam Indragiri," *J. Sist. Inf.*, vol. 2, no. 6, pp. 409–420, 2024. https://jurnal.nawansa.com/index.php/teknofile/article/view/251