Contents list avaliable at Directory of Open Access Journals (DOAJ)

JUTIN : Jurnal Teknik Industri Terintegrasi

Volume 8 Issue 4 2025, Page 3848-3861

ISSN: 2620-8962 (Online)

Journal Homepage: https://journal.universitaspahlawan.ac.id/index.php/jutin/index

Analisis Kerentanan Keamanan pada *Website* Kelurahan Rimba Sekampung dengan Menggunakan *Framework* OWASP ZAP

Nurasmawati[1]✉, Mansur[1], Nurmi Hidayasari[1]

[(1)]Information System Security, Politeknik Negeri Bengkalis, Indonesia

DOI: 10.31004/jutin.v8i4.48523

✉ Corresponding author:
[rnasmawati701@gmail.com]

Article Info

Kata kunci:
Keamanan Website;
OWASP ZAP;
Kerentanan;
Penetration Testing;
Security Assessment

Abstrak

Keamanan aplikasi berbasis web menjadi hal krusial seiring meningkatnya ancaman siber. Penelitian ini menganalisis kerentanan pada website Kelurahan Rimba Sekampung menggunakan OWASP ZAP untuk mengidentifikasi celah keamanan dan memberikan solusi mitigasi. Metode yang digunakan meliputi pemindaian otomatis, analisis hasil, dan implementasi perbaikan. Hasil pemindaian awal menemukan 15 kerentanan yang tergolong dalam kategori Broken Access Control, Security Misconfiguration, Cryptographic Failures, Vulnerable and Outdated Components, serta Software and Data Integrity Failures. Setelah dilakukan perbaikan dan pengujian ulang, jumlah kerentanan berkurang menjadi 12. Solusi yang diterapkan meliputi penguatan Content Security Policy (CSP), penggunaan enkripsi yang lebih kuat, serta konfigurasi cookie dan header HTTP yang sesuai. Implementasi ini terbukti efektif dalam mengurangi risiko keamanan sistem.

Keywords:
Website Security;
OWASP ZAP;
Vulnerability;
Penetration Testing;
Security Assessment

Abstract

The security of web-based applications is increasingly important due to evolving cyber threats. This study analyzes the security vulnerabilities of the Kelurahan Rimba Sekampung website using the OWASP ZAP tool to identify potential weaknesses and recommend mitigation strategies. The methodology includes automated scanning, vulnerability analysis, and applying security improvements. The initial scan identified 15 vulnerabilities, including issues under Broken Access Control, Security Misconfiguration, Cryptographic Failures, Use of Vulnerable and Outdated Components, and Software and Data Integrity Failures. After implementing mitigation measures, the number of vulnerabilities was reduced to 12. Key improvements included strengthening the Content Security Policy (CSP), enhancing encryption mechanisms, and configuring HTTP headers and cookies correctly. These actions significantly reduced the website's security risks. The results

Received 21 July 2025; Received in revised form 28 July 2025 year; Accepted 16 August 2025
Available online 2 October 2025 /

of this study can serve as a reference for web administrators in enhancing application security and safeguarding user data.

1. PENDAHULUAN

Pengujian pada sistem keamanan berbasis *website* adalah hal yang penting pada era perkembangan aplikasi berbasis *website* yang begitu pesat dari masa ke masa. Semakin melaju pesat berkembang aplikasi yang berbasis *website* ini diiringi dengan serangan keamanan yang tinggi dari berbagai teknik ancaman. Sering kali pada masalah keamanan ada pada urutan kedua, atau bisa berada diurutan terakhir dalam hal-hal yang sangat dianggap penting (Edy Listartha et al., 2022). Penggunaan *website* ini adalah salah satu bentuk dari canggihnya penggunaan teknologi masa kini. Oleh karena itu, sangat penting bagi organisasi untuk melakukan *assessment* pada aplikasi yang berbasis *website* agar organisasi mampu mendeteksi kerentanan keamanan dan memahami risiko yang akan dihadapi. Dengan adanya *website* desa warga dapat dengan mudah mengakses informasi mengenai layanan desa tanpa warga datang ke kantor desa untuk layanan yang akan di lakukan dan mendapatkan informasi desa.

Cara yang paling andal untuk menilai sikap keamanan informasi suatu organisasi adalah melalui observasi terhadap responsnya terhadap serangan. Untuk memastikan keamanan sistem secara menyeluruh, melakukan pengujian penetrasi menjadi strategi yang paling terbaik, dimana sering kali memungkinkan para peneliti yang menganalisis keamanan akan menemukan celah baru (Riandhanu, 2022). Analisis *website* kelurahan Rimba Sekampung ini dilakukan untuk menemukan kerentanan keamanan pada *website* untuk mengidentifikasi dampak negatif yang dihasilkan dari analisis kerentanan keamanan.

Sebelum peneliti melaksanakan pengujian keamanan aplikasi berbasis *website* ini, peneliti melakukan tinjauan literatur dan menemukan beberapa penelitian terkait (Edy Listartha et al., 2022). Mitha dalam penelitiannya mendeteksi kerentanan keamanan pada aplikasi berbasis *website* menggunakan OWASP untuk melakukan penilaian risiko agar hasil penelitian mereka diharapkan dapat membantu pihak pengelola dan pengembang sistem untuk dapat mencegah dan mengatasi risiko yang ditemukan pada keamanan sistem informasi *website* tersebut (Edy Listartha et al., 2022). Tamsir dalam penelitiannya menganalisis kerentanan keamanan pada Sistem Informasi Akademik Universitas Bina Darma dengan menggunakan OWASP. Penelitian terdahulu menunjukkan adanya kerentanan keamanan seperti *Cross-Site Scripting* (XSS), *Cross-Site Request Forgery* (CSRF), dan SQL Injection. Berdasarkan analisis OWASP *Risk Rating*, ketiga kerentanan tersebut memiliki tingkat risiko yang berbeda-beda, mulai dari risiko sedang hingga tinggi. Rekomendasi perbaikan dan mitigasi diusulkan untuk meningkatkan keamanan Sistem Informasi Akademik Universitas Bina Darma (Ariyadi et al., 2023). Nurholis dalam penelitiannya ini melakukan analisis kerentanan keamanan *website* Dinas Tenaga Kerja menggunakan metode OWASP *(Open Web Application Security Project*) pada Dinas Tenaga Kerja untuk membantu pihak pengelola dan pengembang sistem mendeteksi dan mengatasi risiko keamanan yang ditemukan pada aplikasi berbasis *website* yang dibangun DisnakerTrans (Aryanti & Utamajaya, 2021).

OWASP *(Open Web Application Security Project)* Top 10 merupakan sebuah metodologi yang digunakan untuk menguji keamanan sistem berbasis *website*. Metode ini dikembangkan oleh komunitas OWASP dan berisi daftar 10 ancaman keamanan utama yang dapat mempengaruhi keamanan suatu situs *website* (Riandhanu, 2022). Metode ini memprioritaskan 10 ancaman tersebut berdasarkan beberapa faktor, yaitu tingkat kemudahan dalam melakukan eksploitasi, seberapa umum atau sering ditemukan ancaman tersebut, kemudahan dalam mendeteksi adanya ancaman, serta dampak parah yang dapat ditimbulkan oleh ancaman tersebut. Dengan menggunakan OWASP Top 10, para pengembang atau penguji keamanan akan dapat mengidentifikasi dan memitigasi kerentanan keamanan kritis yang mungkin ada pada aplikasi *website* mereka ini, sehingga metode ini menjadi acuan penting dalam memastikan keamanan sistem berbasis *website* (Riandhanu, 2022).

Hasil analisis kerentanan yang didapatkan akan membantu pengelola dan pengembang sistem dalam mencegah dan mengatasi dampak risiko yang teridentifikasi pada sistem (Aryanti & Utamajaya, 2021). Belum adanya analisis *Security Assesment* (Penilaian Kemananan) pada sistem informasi *website* kelurahan rimba sekampung. Analisis perlu dilakukan pada keamanan *website* untuk mencegah serangan injeksi agar mencegah kebocoran informasi sensitif, seperti surat dan data masyarakat. Namun belum melakukan pengujian sistem sehingga belum mengetahui secara pasti celah keamanan sistem yang sudah dibangun. Oleh karena itu perlu adanya *Security Assessment* (penilaian keamanan) pada sistem tersebut.

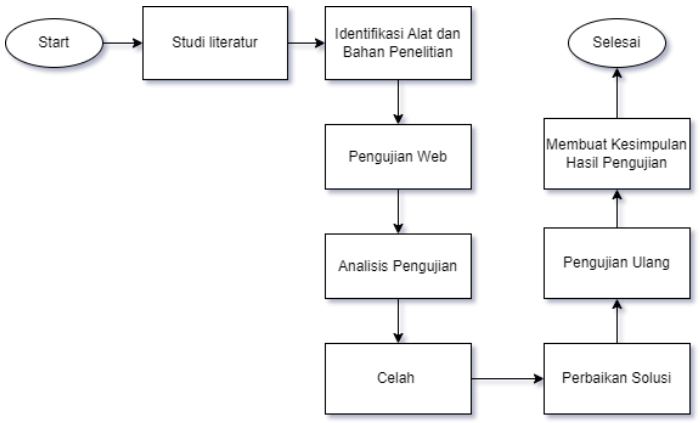
Dalam penggunaan OWASP untuk menguji kerentanan aplikasi website, selain menggunakan alat pemindai kerentanan, diperlukan juga konfigurasi yang tepat untuk memaksimalkan tingkat akurasi hasil pengujian. Konfigurasi yang tepat pada alat pemindai kerentanan dapat membantu mengoptimalkan deteksi kerentanan

yang ada, sehingga proses pengujian dapat memberikan hasil yang lebih komprehensif dan akurat. Hal ini penting dilakukan agar peneliti atau pengembang atau tim keamanan dapat memperoleh informasi yang akurat terkait kelemahan kelemahan keamanan pada aplikasi website yang sedang diuji, sehingga dapat dilakukan tindakan perbaikan yang tepat sasaran (Riandhanu, 2022).

Dalam penelitian ini, terdapat beberapa metode yang dapat digunakan untuk mendeteksi kerentanan keamanan pada aplikasi website, di antaranya ISSAF, OSSTMM, OWASP, dan NIST (OWASP, n.d.). Namun, menurut OWASP (n.d.), metode ini dianggap paling tepat untuk melakukan pengujian penetrasi karena menyediakan panduan standar yang komprehensif serta didukung oleh komunitas global. OWASP, yang didirikan pada tahun 2004, merupakan organisasi nirlaba di Amerika Serikat yang berfokus pada peningkatan keamanan perangkat lunak dan pengembangan pedoman keamanan aplikasi web.

2. METODE

Metode penelitian yang di lakukan untuk menyelesaikan penelitian ini menggunakan desain sistem dengan penjelasan teknikal dari solusi yang berisi urutan-urutan proses yang akan dilakukan untuk menyelesaikan masalah pada penelitian yang akan diteliti.



Gambar 1. Desain Alur Penelitian

2.1 Studi Literatur

Tahap studi literatur digunakan untuk mengumpulkan bahan-bahan yang akan digunakan untuk bantuan pada proses penelitian, seperti pengumpulan jurnal dan artikel tentang website, mempelajari buku-buku, e-book, dan pengumpulan daftar pustaka juga serangkaian kegiatan yang berkaitan dengan metode pengumpulan data pustaka, sehingga penulis menyelesaikan penelitian. Tujuannya adalah agar peneliti bisa memahami teori dan teknik yang sudah terbukti berhasil.

2.2 Identifikasi Alat dan Bahan Penelitian

Tahap identifikasi alat dan bahan digunakan untuk mengetahui alat dan bahan yang akan digunakan dalam pengujian analisis keamanan *website* kelurahan Rimba Sekampung. Kebutuhan alat dan bahan meliputi perangkat keras dan perangkat lunak.

1. Perangkat Keras

   Perangkat keras yang akan digunakan dalam penelitian ini adalah laptop dengan Sistem Operasi *Windows 11 Home Single Language* dengan *Processor* AMD Athlon Silver 3050U *with Radeon Graphics* 2.30 GHz dengan *System Type* 64-bit *operating system*, x64-*based processor*.

2. Perangkat Lunak

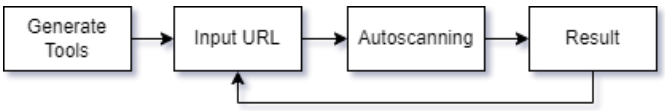
   Perangkat lunak yang akan digunakan dalam penelitian ini adalah *Open Web Application Security Project* (OWASP) versi 2.16.0 dengan kategori pengunduhan *Windows* (64) 228 MB.

2.3 Pengujian Website

Tahap pengujian *website* menggunakan perangkat lunak *Open Web Application Security Project* (OWASP) dan *OWASP Zed Attack Proxy* (ZAP) yang tahapan pengujian ditunjukkan pada Gambar2.

Website yang diuji adalah *website* kelurahan rimba sekampung yang merupakan *website* resmi yang berfungsi sebagai portal informasi dan layanan publik bagi masyarakat kelurahan. *Website* ini menyediakan berbagai

informasi terkait profil kelurahan, lembaga masyarakat, publikasi, dan layanan administrasi seperti surat *online* serta pengaduan masyarakat.



Gambar 2. Alur Pengujian

Hasil pada pengujian akan diperoleh dari hasil uji yang sudah dilakukan, hasil uji yang didapatkan ini merupakah kerentanan-kerentanan yang ada pada pengujian keamanan *website* kelurahan Rimba Sekampung.

2.4 Analisa Pengujian

Tahap analisa hasil pengujian yang di dapat dilakukan setelah hasil melakukan pengujian yang telah di peroleh dimana tahapan ini melalui beberapa tahapan yaitu identifikasi masalah, dampak dan keparahan resiko pada *website,* dan menyesuaikan pemeringkatan resiko (Edy Listartha et al., 2022). Risiko terdapat 5 *level* kategori risiko *level* ini yaitu *Risk, High, Medium, Low*, dan *Informational*. Dengan menelaah hasil pengujian untuk menemukan celah atau pola keamanan yang perlu diperbaiki nantinya pada *website*.

2.5 Celah

Tahap mengidentifikasi dan mendokumentasi kerentanan yang ditemukan selama dianalisis oleh peneliti untuk dilakukan perbaikan celah yang bisa terbuka dari serangan.

2.6 Perbaikan Solusi

Tahap menerapkan solusi untuk mengatasi celah yang sudah ditemukan. Solusi yang diberikan melalui analisa pada pengujian untuk pencegahan kerentanan yang ada pada *website* terdeteksi oleh OWASP ZAP. Solusi yang diberikan nantinya akan disesuaikan dengan kebutuhan kerentanan yang didapat pada saat melakukan penelitian.

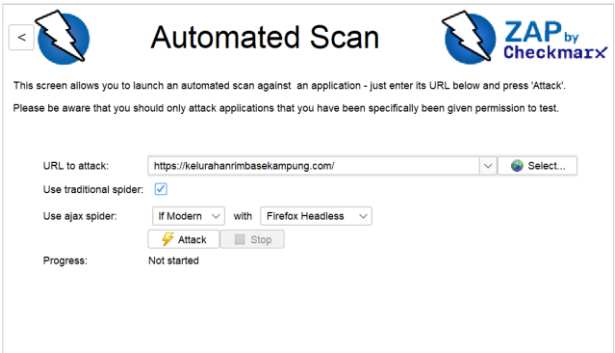
2.7 Pengujian Ulang

Tahap ini dilakukan untuk pengujian kembali menggunakan OWASP ZAP setelah perbaikan solusi yang sudah dilakukan untuk memastikan celah telah tertutup. Pengujian ini untuk memastikan bahwa solusi yang telah diterapkan efektif dan kerentanan sebelumnya telah tertutup. Proses ini melibatkan bahwa celah tidak lagi dapat dieksploitasi, serta memeriksa perbaikan ini bisa mempengaruhi bagian lain dari *website* untuk memvalidasi keseluruhan agar memastikan bahwa seluruh sistem berfungsi dengan baik setelah perubahan. Dengan melakukan tahapan ini memungkinkan peneliti menguji ulang dapat memastikan kualitas dan keamanan sistem sebelum diterapkan secara penuh.

3. HASIL DAN PEMBAHASAN

3.1 Pengujian Website

Pengujian dilakukan dengan menggunakan metode *penetration testing* menggunakan *OWASP zed attack proxy (zap)*. Langkah yang dilakukan yaitu melakukan *generate OWASP zap* dengan cara menghubungkan perangkat *windows OWASP ke Mozilla Firefox*. Selanjutnya menginputkan *url* https://www.kelurahanrimbasekampung.com/, *website* yang akan di *scan* untuk mencari celah kerentanan keamanan *website* yang di tunjukkan pada Gambar 3.



Gambar 3. *Autoscanning* OWASP

Hasil pengujian kerentanan *website* kelurahan rimba sekampung menggunakan *OWASP ZAP* dengan proses *autoscanning* mendapatkan 15 kerentanan yang dapat ditunjukkan pada Gambar 4.

Alert type	Risk	Count
Cloud Metadata Potentially Exposed	High	1 (6.7%)
Content Security Policy (CSP) Header Not Set	Medium	56 (373.3%)
Hidden File Found	Medium	4 (26.7%)
Missing Anti-clickjacking Header	Medium	53 (353.3%)
Cookie No HttpOnly Flag	Low	53 (353.3%)
Cookie Without Secure Flag	Low	53 (353.3%)
Strict-Transport-Security Header Not Set	Low	2 (13.3%)
Timestamp Disclosure - Unix	Low	3 (20.0%)
X-Content-Type-Options Header Missing	Low	220 (1,466.7%)
Information Disclosure - Suspicious Comments	Informational	66 (440.0%)
Modern Web Application	Informational	53 (353.3%)
Re-examine Cache-control Directives	Informational	1 (6.7%)
Session Management Response Identified	Informational	60 (400.0%)
User Agent Fuzzer	Informational	12 (80.0%)
Vulnerable JS Library	Informational	1 (6.7%)
Total		15

Gambar 4. Tabel Laporan Pengujian *Autoscanning*

3.2 Analisa Pengujian

Dalam melakukan analisis keamanan pada aplikasi yang berbasis *website*, hasil pemindaian dengan menggunakan alat *OWASP ZAP* memberikan kontribusi peran yang penting dalam mengidentifikasi potensi kerentanan. Dengan mengacu pada kategori *OWASP* Top 10 tahun 2021, temuan dari hasil pemindaian ini dapat dikelompokkan sesuai dengan jenis kerentanannya. Kategori *OWASP* Top 10 menyediakan kerangka kerja standar yang digunakan secara luas dalam industri keamanan *siber* untuk memahami dan menangani risiko keamanan aplikasi. Oleh karena itu, tabel berikut menyajikan detail *alert* yang teridentifikasi selama pemindaian *OWASP ZAP*, dikaitkan dengan kategori *OWASP* Top 10, beserta penjelasan mengenai kerentanannya. Informasi ini diharapkan dapat membantu dalam mengidentifikasi area yang memerlukan tindakan mitigasi untuk meningkatkan keamanan pada aplikasi *website*.

Tabel 1. Detail *Alert OWASP ZAP* dengan kategori *OWASP* Top 10 2021

No	Alert	Kategori OWASP Top 10 (2021)	Kerentanan
1	Cloud Metadata Potentially Exposed	A01:2021 - Broken Access Control	Potensi akses metadata yang tidak sah.
2	Content Security Policy (CSP) Header Not Set	A05:2021 - Security Misconfiguration	Tidak adanya header CSP memungkinkan serangan seperti XSS.
3	Hidden File Found	A05:2021 - Security Misconfiguration	File tersembunyi dapat dieksploitasi jika mengandung informasi sensitif.
4	Missing Anti-clickjacking Header	A05:2021 - Security Misconfiguration	Tidak adanya header memungkinkan serangan clickjacking.

No	Alert	Kategori OWASP Top 10 (2021)	Kerentanan
5	Cookie No HttpOnly Flag	A05:2021 - Security Misconfiguration	Cookie dapat diakses melalui skrip sisi klien, meningkatkan risiko pencurian.
6	Cookie Without Secure Flag	A05:2021 - Security Misconfiguration	Cookie dikirim melalui koneksi tidak terenkripsi, memungkinkan pencurian.
7	Strict-Transport-Security Header Not Set	A05:2021 - Security Misconfiguration	Tidak adanya HSTS memungkinkan serangan downgrade dan sniffing.
8	Timestamp Disclosure - Unix	A05:2021 - Security Misconfiguration	Informasi timestamp yang terekspos dapat memberikan petunjuk pada penyerang.
9	X-Content-Type-Options Header Missing	A05:2021 - Security Misconfiguration	Tidak adanya header memungkinkan serangan MIME sniffing.
10	Information Disclosure - Suspicious Comments	A05:2021 - Security Misconfiguration	Komentar kode yang terekspos dapat mengungkapkan informasi tentang aplikasi.
11	Modern Web Application	A08:2021 - Software and Data Integrity Failures	Aplikasi modern dapat memiliki risiko integrasi yang tidak aman.
12	Re-examine Cache-control Directives	A05:2021 - Security Misconfiguration	Cache-control tidak diatur dengan benar, memungkinkan data sensitif di-cache.
13	Session Management Response Identified	A02:2021 - Cryptographic Failures	Manajemen sesi tidak aman dapat meningkatkan risiko serangan sesi.
14	User Agent Fuzzer	A05:2021 - Security Misconfiguration	User agent digunakan untuk mendeteksi kelemahan pada aplikasi.
15	Vulnerable JS Library	A06:2021 - Vulnerable and Outdated Components	Pustaka JS yang rentan memungkinkan eksploitasi kerentanan yang diketahui.

3.3 Celah

Berdasarkan hasil pengujian keamanan yang telah dilakukan, ditemukan sejumlah kerentanan pada *website* milik kelurahan rimba sekampung. Kerentanan-kerentanan ini dapat menjadi pintu masuk bagi pihak tidak bertanggung jawab untuk mengeksploitasi sistem dan mengganggu operasional *website*. Proses pengujian mendapatkan 15 kerentanan yang di tunjukkan pada tabel 2.

Tabel 2. Celah Kerentanan

No	Alert	Risk Level	Skript	CWE ID	WASC ID
1	Cloud Metadata Potentially Exposed	High	1	0	0
2	Content Security Policy (CSP) Header Not Set	Medium	52	693	15
3	Hidden File Found	Medium	4	538	13
4	Missing Anti-clickjacking Header	Medium	50	1021	15
5	Cookie No HttpOnly Flag	Low	50	1004	13
6	Cookie Without Secure Flag	Low	50	614	13
7	Strict-Transport-Security Header Not Set	Low	1	319	15
8	Timestamp Disclosure - Unix	Low	3	200	13
9	X-Content-Type-Options Header Missing	Low	212	693	15
10	Information Disclosure - Suspicious Comments	Informational	63	200	13
11	Modern Web Application	Informational	50	-	-
12	Re-examine Cache-control Directives	Informational	1	525	13
13	Session Management Response Identified	Informational	59	-	-
14	User Agent Fuzzer	Informational	12	0	0
15	Vulnerable JS Library	Informational	1	829	-

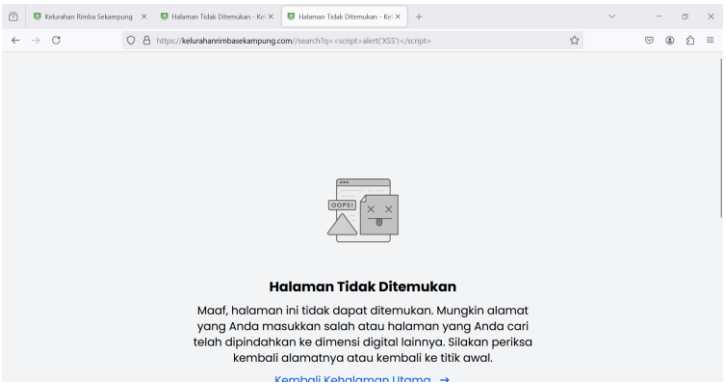
Langkah *Penetration Testing* pada tahapan ini penulis akan melakukan berbagai percobaan dalam pengujian kerentanan keamanan *website* kelurahan rimba sekampung dengan menggunakan berbagai cara dan teknik berdasarkan informasi *scanning* sebelum melakukan teknik serangan random.

a) Teknik XSS (Cross-Site Scripting)

Hasil pengujian menunjukkan bahwa tidak ditemukan refleksi input, yang berarti aplikasi telah memproses input dengan baik sehingga mencegah serangan Reflected XSS.

Gambar 5. Pengujian Reflected XSS menggunakan XSStrike pada Parameter URL

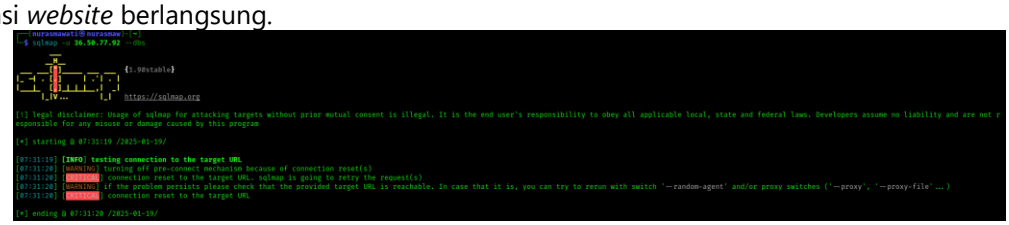
Pengujian dengan teknik *xss (cross-site scripting)* peneliti akan menggunakan beberapa *script* untuk mencoba melakukan serangan xss manual ke dalam *website* kelurahan rimba sekampung https://kelurahanrimbasekampung.com/. Dengan menyelipkan *script* pada *link url website* kelurahan rimba sekampung seperti berikut. Pengujian dengan serangan *xss (cross-site scripting)* fungsi dari *payload* dasar untuk *XSS Reflected payload* ini yaitu untuk memunculkan *popup alert* di *browser* dengan pesan "*XSS*" tanpa merusak sistem yang akan di uji.



Gambar 6. Percobaan *Skrip XSS* Manual

b) Teknik SQL Injection

Dengan menggunakan *SQLMAP*, peneliti dapat mengidentifikasi celah keamanan dalam mekanisme *query* basis data dan menyebarkan potensi risiko yang mungkin timbul jika kerentanan ini dieksploitasi. Proses ini dilakukan secara terstruktur dan terkendali untuk memastikan integritas sistem tetap terjaga selama pengujian keamanan aplikasi *website* berlangsung.



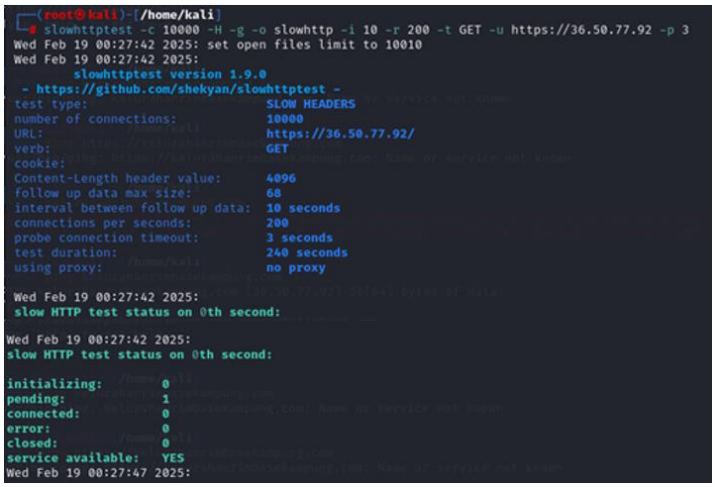
Gambar 7. Hasil Pengujian *SQL Injection*

Berikut gambaran ini menunjukkan akses daftar *database* pada *server* target, namun proses tersebut gagal karena koneksi terputus secara paksa oleh *server*. Hal ini kemungkinan disebabkan oleh adanya mekanisme keamanan seperti *Web Application Firewall (WAF)* atau sistem *Intrusion Detection/Prevention System (IDS/IPS)* yang mendeteksi dan memblokir permintaan mencurigakan. Selain itu, kegagalan ini juga terjadi jika *server* tidak responsif, memblokir *IP* pengguna, atau tidak memiliki kerentanan *SQL Injection*.

c) Serangan DoS (Denial of Service)

Peneliti melakukan serangan DoS dengan *tools* SlowHTTPTest dalam pengujian ini, metode yang digunakan adalah *Slow Headers Attack*, di mana server dikirimkan header HTTP secara perlahan agar koneksi tetap terbuka

dan mempengaruhi kinerja server. Pengujian dilakukan dengan 10.000 koneksi, setiap koneksi mengirimkan data dalam interval 10 detik, dengan laju koneksi baru sebanyak 200 per detik, dapat dilihat pada gambar.



Gambar 7. Hasil Pengujian Slow HTTP Attack dengan SlowHTTPTest

3.4 Perbaikan Solusi

Solusi yang diusulkan oleh penulis dalam penelitian ini dirancang untuk membantu pengelola *website* dalam mengidentifikasi dan mencegah kerentanan yang terdeteksi menggunakan OWASP ZAP. Dengan mengacu pada modul *CWE (Common Weakness Enumeration)*, solusi ini bertujuan untuk mengurangi risiko kerentanan yang ada dan secara proaktif meningkatkan tingkat keamanan *website* kelurahan rimba sekampung. Implementasi langkah-langkah ini oleh pengelola *website* diharapkan dapat menciptakan lingkungan digital yang lebih aman dan andal.

Tabel solusi dibawah ini adalah sebelum penyerahan solusi ke pengelola *website* kelurahan rimba sekampung yang dimulai pada tanggal 23 Januari 2025.

Tabel 3. Solusi dari kerentanan *OWASP ZAP*

No	Alert	Risk Level	Solusi OWASP ZAP
1	Cloud Metadata Potentially Exposed	High	Hindari penggunaan *$host* langsung di NGINX. Gunakan *$server_name* atau bersihkan *$host* untuk mencegah serangan.
2	Content Security Policy (CSP) Header Not Set	Medium	Pastikan header CSP diset pada level server (baik itu server web, aplikasi, atau load balancer) sehingga browser dapat menerapkannya dan memitigasi risiko yang terkait dengan konten yang tidak aman.
3	Hidden File Found	Medium	Pertimbangkan apakah komponen tersebut benar-benar diperlukan dalam produksi, jika tidak, nonaktifkan saja. Jika ya, pastikan akses ke komponen tersebut memerlukan autentikasi dan otorisasi yang sesuai, atau batasi paparan ke sistem internal atau IP sumber tertentu, dll.
4	Missing Anti-clickjacking Header	Medium	Pastikan untuk mengatur header *X-Frame-Options* atau menggunakan direktif *frame-ancestors* dalam Content-Security-Policy (CSP) pada semua halaman web yang dikembalikan oleh situs web. Jika halaman web seharusnya hanya dimuat dalam frame dari halaman yang ada di server (misalnya, bagian dari FRAMESET), maka gunakan pengaturan SAMEORIGIN. Namun, jika tidak menginginkan halaman dimuat dalam frame sama sekali, disarankan untuk menggunakan pengaturan DENY. Sebagai alternatif tambahan dapat menggunakan direktif frame-ancestors dalam CSP untuk memberikan kontrol yang lebih terperinci mengenai siapa yang diperbolehkan untuk menyematkan halaman dalam frame.
5	Cookie No HttpOnly Flag	Low	Pastikan bahwa tanda *HttpOnly* diatur pada semua cookie yang digunakan oleh situs web.
6	Cookie Without Secure Flag	Low	Pastikan bahwa semua cookie yang mengandung informasi sensitif, seperti session token, dikirim melalui koneksi yang terenkripsi menggunakan HTTPS. Hal ini dilakukan dengan mengaktifkan atribut Secure pada cookie, sehingga browser hanya akan mengirim cookie tersebut melalui koneksi aman. Selain itu,

No	Alert	Risk Level	Solusi OWASP ZAP
			konfigurasi server atau aplikasi web perlu diperbarui untuk memastikan semua cookie sensitif diatur dengan atribut Secure.
7	Strict-Transport-Security Header Not Set	Low	Pastikan bahwa server web, aplikasi server, atau load balancer dikonfigurasi untuk menerapkan header Strict-Transport-Security (HSTS). Header ini memaksa browser hanya menggunakan koneksi HTTPS untuk mengakses situs, sehingga mencegah serangan downgrade dari HTTPS ke HTTP serta melindungi dari serangan Man-in-the-Middle (MITM). Untuk mengimplementasikannya, tambahkan konfigurasi HSTS pada server dengan nilai seperti *max-age=31536000; includeSubDomains*, yang menginstruksikan browser untuk hanya menggunakan HTTPS selama satu tahun penuh, termasuk pada semua subdomain.
8	Timestamp Disclosure - Unix	Low	Konfirmasikan secara manual untuk memastikan bahwa data timestamp yang diungkapkan tidak bersifat sensitif dan tidak dapat digunakan untuk mengungkap pola yang dapat dieksploitasi. Jika timestamp diperlukan untuk keperluan operasional, pastikan bahwa data tersebut dienkripsi, dibatasi aksesnya, atau hanya diungkapkan jika benar-benar diperlukan, guna meminimalkan risiko.
9	X-Content-Type-Options Header Missing	Low	Pastikan bahwa aplikasi atau server web selalu menyetel header Content-Type dengan tepat sesuai jenis file yang dikirimkan. Selain itu, tambahkan header X-Content-Type-Options dengan nilai 'nosniff' pada semua halaman web. Header ini mencegah browser melakukan MIME-sniffing, yang dapat digunakan oleh penyerang untuk mengelabui browser dalam mengeksekusi file berbahaya. Jika memungkinkan, pastikan pengguna menggunakan browser modern yang mematuhi standar dan tidak melakukan MIME-sniffing, atau dapat diarahkan oleh aplikasi atau server untuk tidak melakukannya. Hal ini meningkatkan perlindungan terhadap serangan seperti MIME type confusion dan mengurangi risiko eksekusi konten tidak aman.
10	Information Disclosure - Suspicious Comments	Informational	Hapus semua komentar yang mengembalikan informasi yang dapat membantu penyerang dan perbaiki masalah mendasar yang mereka rujuk.
11	Modern Web Application	Informational	Ini adalah peringatan informasional sehingga tidak diperlukan perubahan atau tindakan apa pun.
12	Re-examine Cache-control Directives	Informational	Pastikan bahwa header cache-control diatur dengan benar sesuai kebutuhan keamanan dan caching konten. Untuk konten yang bersifat sensitif, gunakan nilai "*no-cache, no-store, must-revalidate*" untuk mencegah penyimpanan atau penggunaan ulang data yang di-cache. Jika aset tertentu perlu di-cache, gunakan nilai seperti "*public, max-age, immutable*", yang memungkinkan caching dengan durasi yang ditentukan sambil memastikan konten tidak berubah. Konfigurasi ini membantu mengoptimalkan caching tanpa mengorbankan keamanan.
13	Session Management Response Identified	Informational	Ini adalah peringatan informasional dan bukan kerentanan sehingga tidak ada yang perlu diperbaiki.
14	User Agent Fuzzer	Informational	Pastikan aplikasi tidak memberikan respon spesifik yang berpotensi digunakan oleh penyerang, seperti untuk crawler mesin pencari atau perangkat tertentu. Selain itu, validasi semua permintaan pada sisi server untuk memastikan hanya permintaan yang sah yang diproses, tanpa memandang user agent yang digunakan. Langkah ini dapat dilengkapi dengan memantau log server untuk aktivitas mencurigakan dan mengimplementasikan Web Application Firewall (WAF) untuk mendeteksi serta memblokir potensi serangan.
15	Vulnerable JS Library	Informational	Harap tingkatkan ke versi terbaru chart.js.

3.5 Pengujian Ulang

Pengujian ulang yang dilakukan pada 4 Februari 2025 menghasilkan 12 *alert* yang terdeteksi oleh *OWASP ZAP*. Dengan membandingkan hasil ini dengan pemindaian sebelumnya, di sini peneliti dapat melakukan analisis terkait perbedaan untuk membandingkan dari pengujian awal ke pengujian ulang.

Tabel 4. Hasil Pemindaian Ulang Kerentanan yang Masih Tersisa

No	Alert	Risk Level	Deteksi Pemindaian Ulang
1	Cloud Metadata Potentially Exposed	High	Tidak ditemukan
2	Content Security Policy (CSP) Header Not Set	Medium	Ditemukan
3	Hidden File Found	Medium	Tidak ditemukan
4	Missing Anti-clickjacking Header	Medium	Ditemukan
5	Cookie No HttpOnly Flag	Low	Ditemukan
6	Cookie Without Secure Flag	Low	Ditemukan
7	Strict-Transport-Security Header Not Set	Low	Ditemukan
8	Timestamp Disclosure - Unix	Low	Ditemukan
9	X-Content-Type-Options Header Missing	Low	Ditemukan
10	Information Disclosure - Suspicious Comments	Informational	Ditemukan
11	Modern Web Application	Informational	Ditemukan
12	Re-examine Cache-control Directives	Informational	Ditemukan
13	Session Management Response Identified	Informational	Ditemukan
14	User Agent Fuzzer	Informational	Ditemukan
15	Vulnerable JS Library	Informational	Tidak ditemukan.

Analisis data hasil kerentanan ini dari pengujian ulang yaitu 12 kerentanan yang sudah dilakukan peneliti merekomendasi untuk meminimalkan risiko keamanan yang ditemukan selama pengujian ulang, sehingga sistem menjadi lebih aman dan mampu mengantisipasi potensi serangan siber. Setiap celah keamanan memiliki dampak yang beragam, mulai dari kebocoran data pengguna, potensi penyisipan skrip berbahaya, hingga kemungkinan eksploitasi oleh pihak tidak bertanggung jawab melalui metode tertentu. Oleh karena itu, langkah-langkah perbaikan yang disarankan mencakup penerapan kebijakan keamanan yang lebih ketat, pengaturan ulang konfigurasi header HTTP yang relevan, serta penguatan sistem manajemen sesi dan perlindungan data. Dengan mengimplementasikan rekomendasi ini, diharapkan website dapat mengurangi kerentanan terhadap ancaman siber dan meningkatkan perlindungan terhadap informasi sensitif pengguna.

Tabel 5. Dampak dan Rekomendasi

No	Kerentanan	Dampak	Rekomendasi
1	Content Security Policy (CSP) Header Not Set	Website rentan terhadap serangan Cross-Site Scripting (XSS) yang memungkinkan peretas menyisipkan skrip berbahaya untuk mencuri data pengguna.	Tambahkan Content Security Policy (CSP) pada konfigurasi server untuk membatasi sumber daya eksternal yang dapat dijalankan di website.
2	Missing Anti-clickjacking Header	Website bisa dimasukkan dalam halaman lain secara tersembunyi, memungkinkan serangan clickjacking, di mana pengguna tanpa sadar melakukan tindakan yang tidak mereka maksudkan.	Tambahkan header X-Frame-Options: DENY atau gunakan kebijakan keamanan Content-Security-Policy frame-ancestors untuk mencegah tampilan dalam iframe.
3	Cookie No HttpOnly Flag	Cookie yang menyimpan data sesi pengguna bisa diakses oleh skrip berbahaya, yang memungkinkan peretas mencuri informasi sesi dan mengambil alih akun.	Pastikan semua cookie yang menyimpan data sensitif menggunakan atribut HttpOnly agar tidak dapat diakses oleh skrip di halaman web.
4	Cookie Without Secure Flag	Cookie dapat dikirim melalui koneksi tidak aman (HTTP), sehingga rentan terhadap serangan Man-in-the-Middle (MITM) yang dapat mencuri data pengguna.	Gunakan atribut Secure pada cookie agar hanya dikirim melalui koneksi HTTPS yang terenkripsi.
5	Strict-Transport-Security Header Not Set	Pengguna dapat mengakses website melalui HTTP yang tidak aman, memungkinkan serangan downgrade attack atau Man-in-the-Middle (MITM).	Terapkan HTTP Strict Transport Security (HSTS) agar semua permintaan website selalu diarahkan ke HTTPS.

No	Kerentanan	Dampak	Rekomendasi
6	Timestamp Disclosure - Unix	Website membocorkan informasi waktu dalam format Unix Timestamp, yang bisa digunakan peretas untuk menganalisis pola aktivitas sistem dan merancang serangan lebih akurat.	Sembunyikan timestamp jika tidak diperlukan atau enkripsi data yang mengandung informasi waktu untuk mencegah eksploitasi.
7	X-Content-Type-Options Header Missing	Browser bisa menebak tipe file yang diunduh, yang dapat dieksploitasi untuk mengubah file biasa menjadi file berbahaya.	Tambahkan header X-Content-Type-Options: nosniff agar browser hanya membaca file sesuai tipe aslinya.
8	Information Disclosure - Suspicious Comments	Komentar dalam kode website bisa mengandung informasi sensitif seperti password atau API key, yang bisa digunakan peretas untuk menyerang sistem.	Lakukan audit kode sumber secara rutin dan hapus komentar yang berisi informasi sensitif sebelum dipublikasikan.
9	Modern Web Application	Tidak ada dampak langsung, tetapi fitur modern yang tidak dikonfigurasi dengan baik bisa menjadi celah keamanan yang dapat dimanfaatkan peretas.	Lakukan pengujian keamanan rutin untuk memastikan semua fitur web modern digunakan dengan aman.
10	Re-examine Cache-control Directives	Data sensitif bisa tersimpan di cache browser atau proxy, memungkinkan akses tidak sah meskipun pengguna sudah logout.	Gunakan aturan Cache-Control: no-cache, no-store, must-revalidate untuk memastikan informasi sensitif tidak tersimpan dalam cache.
11	Session Management Response Identified	Informasi tentang bagaimana sesi pengguna dikelola dapat digunakan oleh peretas untuk mencoba membajak sesi atau meniru sesi yang sah.	Pastikan session ID dibuat dengan aman, dikirim hanya melalui HTTPS, dan memiliki masa berlaku terbatas dengan mekanisme session timeout.
12	User Agent Fuzzer	Peretas atau bot otomatis mencoba mengidentifikasi celah keamanan dengan mengubah User-Agent, yang bisa digunakan untuk eksploitasi sistem.	Gunakan Web Application Firewall (WAF) atau validasi di sisi server untuk mendeteksi dan memblokir permintaan mencurigakan.

Peneliti telah melakukan pengujian terhadap website kelurahan rimba sekampung menggunakan berbagai teknik untuk mengidentifikasi potensi kerentanan keamanan. Pengujian ini mencakup teknik Cross-Site Scripting (XSS), SQL Injection, dan serangan Denial of Service (DoS). Dari hasil pengujian yang dilakukan, dapat disimpulkan beberapa hal berikut:

1. Pengujian XSS

   Dalam pengujian ini, peneliti menggunakan alat XSStrike untuk mendeteksi potensi serangan XSS pada halaman search.php dari website kelurahan rimba sekampung. Hasil pengujian menunjukkan bahwa terdapat beberapa objek yang berpotensi rentan, namun XSStrike tidak menemukan payload eksploitasi langsung. Selanjutnya, peneliti melakukan fuzzing parameter menggunakan opsi --fuzzer, tetapi hasilnya menunjukkan bahwa tidak ditemukan refleksi input, yang berarti aplikasi telah memiliki mekanisme pemrosesan input yang baik dalam mencegah serangan Reflected XSS. Ini menandakan bahwa ada sistem sanitasi input atau filter keamanan yang telah diterapkan pada website. Beberapa percobaan manual dengan menyisipkan skrip ke dalam URL juga telah dilakukan, termasuk penggunaan event handler seperti onmouseover untuk mencuri cookie. Namun, website telah menerapkan proteksi seperti Content Security Policy (CSP) dan Web Application Firewall (WAF), sehingga serangan ini tidak berhasil.

2. Pengujian SQL Injection

   Peneliti telah melakukan pengujian SQL Injection menggunakan SQLMAP untuk melihat apakah terdapat celah keamanan dalam mekanisme query database aplikasi. Hasil uji menunjukkan bahwa upaya untuk mengakses daftar database server gagal karena koneksi terputus secara paksa oleh server. Hal ini kemungkinan besar disebabkan oleh adanya mekanisme keamanan seperti WAF atau sistem IDS/IPS yang secara aktif memantau dan memblokir aktivitas mencurigakan. Selain itu, peneliti menggunakan waybackurls untuk mengekstraksi daftar URL yang pernah tersedia di website tersebut, guna mengidentifikasi parameter yang mungkin rentan terhadap SQL Injection. Namun, hasilnya menunjukkan bahwa tidak ada parameter yang dapat dieksploitasi, menandakan bahwa website telah menerapkan teknik keamanan seperti prepared statements atau sanitasi input yang baik.

3. Pengujian Denial of Service (DoS)

Dalam pengujian serangan DoS, peneliti menggunakan SlowHTTPTest dengan metode Slow Headers Attack, di mana server dikirimkan header HTTP secara perlahan agar koneksi tetap terbuka dan membebani server. Pengujian dilakukan dengan 10.000 koneksi, setiap koneksi mengirimkan data dalam interval 10 detik, dengan laju koneksi baru sebanyak 200 per detik. Hasil pengujian menunjukkan bahwa server masih mampu menangani serangan ini, dengan status layanan tetap tersedia (Service Available: YES). Hanya terdapat satu koneksi yang tertunda, dan tidak ditemukan kesalahan dalam pengiriman data. Ini menunjukkan bahwa server memiliki ketahanan terhadap metode serangan yang digunakan.

4. KESIMPULAN

Berdasarkan hasil analisis yang telah dilakukan dalam penelitian mengenai Analisis Kerentanan Keamanan pada *Website* Kelurahan Rimba Sekampung, dapat disimpulkan bahwa pengujian awal menggunakan *OWASP ZAP* berhasil mengidentifikasi 15 kerentanan yang diklasifikasikan berdasarkan *OWASP* Top 10 (2021). Mayoritas kerentanan yang ditemukan berkaitan dengan *Security Misconfiguration*, yang menunjukkan adanya kelemahan dalam konfigurasi keamanan sistem. Setelah dilakukan implementasi solusi mitigasi, jumlah kerentanan berkurang menjadi 12 *alert* pada pengujian ulang, yang membuktikan bahwa langkah perbaikan yang diterapkan cukup efektif dalam menurunkan risiko keamanan. *Alert* yang sudah tidak ditemukan itu adalah *Cloud Metadata Potentially Exposed* level risiko *high, Hidden File Found* level risiko *medium, Vulnerable JS Library* level risiko *informational*. Namun, meskipun terjadi penurunan jumlah *alert*, masih terdapat beberapa kerentanan dengan tingkat risiko sedang yang masih tersisa, serta kerentanan dengan tingkat risiko rendah yang belum mengalami perubahan yang perlu diperbaiki lebih lanjut agar keamanan *website* dapat lebih optimal. Penelitian ini juga menunjukkan bahwa metode *penetration testing* menggunakan *OWASP ZAP* efektif dalam mengidentifikasi serta mengurangi risiko keamanan. Selain itu, penelitian ini memberikan rekomendasi perbaikan yang dapat diterapkan untuk meningkatkan sistem keamanan *website* secara keseluruhan.

Pengujian menggunakan penetrasi testing manual meliputi:

a) Pengujian serangan XSS Tidak Ditemukan

Pengujian terhadap celah Cross-Site Scripting (XSS) menunjukkan bahwa *website* tidak rentan terhadap serangan ini. Saat dilakukan percobaan untuk menyisipkan *script* berbahaya ke dalam URL, sistem secara otomatis menolak permintaan tersebut. Hal ini mengindikasikan bahwa mekanisme keamanan pada *website* telah bekerja dengan baik dalam mencegah eksploitasi XSS.

b) SQL Injection Tidak Ditemukan

Pengujian eksploitasi terhadap SQL Injection juga tidak menemukan adanya celah keamanan yang dapat dimanfaatkan oleh penyerang. Seluruh input yang diuji telah diproses dengan aman, sehingga *website* tidak rentan terhadap manipulasi query database melalui teknik SQL Injection. Ini menunjukkan bahwa sistem telah memiliki perlindungan yang cukup baik terhadap jenis serangan ini.

c) Serangan Denial of Service (DoS)

Pengujian terhadap serangan Denial of Service (DoS) menunjukkan bahwa serangan ini berhasil dilakukan. *Website* mengalami gangguan dan tidak dapat diakses sementara waktu akibat beban lalu lintas yang tinggi yang dikirimkan secara bersamaan. Hasil ini mengindikasikan bahwa sistem masih memiliki kelemahan dalam menangani lonjakan permintaan yang tidak wajar, yang berpotensi dimanfaatkan oleh penyerang untuk menyebabkan downtime pada layanan. Tetapi *website* kelurahan rimba sekampung masih bisa beroperasi seperti biasa tanpa ada kerusakan sistem dari serangan ini.

5. REFERENSI

Ariyadi, T., Widodo, T. L., Apriyanti, N., & Kirana, F. S. (2023). Analisis Kerentanan Keamanan Sistem Informasi Akademik Universitas Bina Darma Menggunakan OWASP. *Techno.Com*, *22*(2), 418–429. https://doi.org/10.33633/tc.v22i2.7562

Aryanti, D., & Utamajaya, J. N. (2021). Analisis Kerentanan Keamanan Website Menggunakan Metode OWASP (Open Web Application Security Project) Pada Dinas Tenaga Kerja. *Jurnal Syntax Fusion*, *1*(03), 15–25.

Edy Listartha, I. M., Premana Mitha, I. M. A., Aditya Arta, M. W., & Yuda Arimika, I. K. W. (2022). Analisis Kerentanan Website SMA Negeri 2 Amlapura Menggunakan Metode OWASP (Open Web Application Security Project). *Simkom*, *7*(1), 23–27. https://doi.org/10.51717/simkom.v7i1.63

Riandhanu, I. O. (2022). Analisis Metode Open Web Application Security Project (OWASP) Menggunakan

Penetration Testing pada Keamanan Website Absensi. *Jurnal Informasi Dan Teknologi*. https://doi.org/10.37034/jidt.v4i3.236

Ghozali, B., Kusrini, K., & Sudarmawan, S. (2019). Mendeteksi kerentanan keamanan aplikasi website menggunakan metode OWASP (Open Web Application Security Project) untuk penilaian risk rating. Creative Information Technology Journal, 4(4), 264–275. https://citec.amikom.ac.id/main/index.php/citec/article/view/119

Tangkudung, I., Dako, R. D. R., & Dako, A. Y. (2019). Evaluasi website menggunakan metode ISO/IEC 25010. In SemanTECH (Seminar Nasional Teknologi, Sains dan Humaniora) (pp. 87–107).

Sinaga, A. S. R. M. (2020). *Keamanan komputer*. CV Insan Cendekia Mandiri.

Zahra, N. A., Zidane, F. H., & Kuslaila, N. R. (2023). Analisis keamanan sistem informasi pada website PT Sentra Vidya Utama (SEVIMA) menggunakan metode OWASP. Prosiding Seminar Nasional Teknologi dan Sistem Informasi, 3(1), 384–393. https://doi.org/10.33005/sitasi.v3i1.564

Hidayatulloh, S., & Saptadiaji, D. (2021). Penetration testing pada website Universitas ARS menggunakan Open Web Application Security Project (OWASP). Jurnal Algoritma, 18(1), 77–86. https://doi.org/10.33364/algoritma/v.18-1.827

Al'am'yubi, M. R. S., & Wijayanto, D. (2023). Analisis sistem keamanan website XYZ menggunakan framework OWASP ZAP. Jurnal Ilmu Komputer, 3(1), 1–5. https://journal.umgo.ac.id/index.php/juik/index

Adinugroho, N. B., Hendradi, P., & Sasongko, D. (2022). Analisis keamanan e-learning menggunakan Open Web Application Security Project (OWASP) (Studi kasus MOCA UNIMMA). Jurnal Informatika, 22(2), 132–138. https://doi.org/10.30873/ji.v22i2.3327

OWASP. (n.d.). About OWASP. Retrieved July 16, 2024, from https://www.owasp.org/index.php/About_OWASP