Vol. 2 No. (1) (Juni 2024) 9-17

Journal of Data Insights

http://journalnew.unimus.ac.id/index.php/jodi

Deteksi Kerentanan SQL Injection pada Website Menggunakan Vulnerability Assessment

Nova Christina Sari[1], **Achmad Solichan**[2], **Basirudin Ansor**[3] , **Aditya Putra Ramdani**[4], **Muhammad Zainudin Al Amin**[5], **Mulil Khaira**[6], **Auliya Rohman Riquelme Al Ubaidah**[7]
[1234567]Teknologi Informasi Universitas Muhammadiyah Semarang, Indonesia

DOI: https://doi.org/10.26714/jodi

Info Artikel

Sejarah Artikel:
Disubmit 6 Juni 2024
Direvisi 16 Juni 2024
Disetujui 30 Juni 2024

Keywords:
Cyber Security, OWASP, SQL Injection, Vulnerability, Zap

Abstrak

Website merupakan sumber informasi yang terus berkembang mengikuti perkembangan teknologi dimasa sekarang, dan penggunaan website telah banyak diterapkan di berbagai bidang. Penggunaan cyber security berkembang sangat pesan mengikuti perkembangan teknologi dan penelitian-penelitian dalam tema cyber security semakin banyak. Metode vulnerability asesement dalam penelitian ini digunakan untuk melakukan scanning terhadap suatu website untuk mendeteksi kerentananan website tersebut terhadap SQL Injection. SQL Injection. SQL Injection merupakan salah satu teknik peretasan dengan cara menyalahgunakan celah keamanan yang ada di lapisan SQL berbasis data pada suatu aplikasi atau website. Vulnerability scanner menggunakan Zap, memberikan hasil 22 alert, 3 diantaranya memiliki kerentanan yang tinggi (high risk). Hasil dari kerentanan website memiliki SQL Injenction yang tinggi. Terdapat 5 dari 10 ancaman yang terdapat pada OWASP Top 10.

Abstract

Websites are a source of information that continues to grow following current technological developments, and the use of websites has been widely applied in various fields. The use of cybersecurity is growing very rapidly following technological developments and research on the theme of cybersecurity is increasing. The vulnerability assessment method in this study is used to scan a website to detect the vulnerability of the website to SQL Injection. SQL Injection. SQL Injection is a hacking technique by exploiting security holes in the SQL-based data layer of an application or website. The vulnerability scanner uses Zap, giving 22 warnings, 3 of which have high vulnerability (high risk). The results of the website vulnerability have high SQL Injection. There are 5 out of 10 threats in the OWASP Top 10

✉ Alamat Korespondensi:
E-mail: novachistinasari@unimus.ac.id

e-ISSN: 2988 - 2109

1. PENDAHULUAN

Website merupakan sumber informasi yang terus berkembang mengikuti perkembangan teknologi dimasa sekarang, dan penggunaan *website* telah banyak diterapkan di berbagai bidang(Riandhanu, 2022). *Website* banyak digunakan sebagai media penyimpanan data, dan informasi yang dapat diakses secara publik maupun secara rahasia(Hardiani et al., 2022). Penyimpanan data pribadi yang terdapat didalam sebuah *website* harus bersifat aman, sehingga *website* tersebut harus memiliki sistem keamanan yang baik dan dapat menjamin keamanan data yang tersimpan pada server *website* tersebut(Fucci et al., 2024).

Penggunaan *cyber security* berkembang sangat pesan mengikuti perkembangan teknologi. Penelitian-penelitian dalam tema *cyber security* semakin banyak. Penelitian *vulnerability assessment* meneliti tentang keamanan *website* menggunakan aplikasi *vulnerability scanner* (Sharma et al., 2023). Penelitian keamanan *website* menggunakan *blackbox* dilakukan untuk memastikan permasalahan yang terjadi dalam *website* tersebut (Aurangzeb et al., 2024). Penelitian dengan OWASP Top 10 sering dilakukan untuk mengetahui kerentanan suatu *website* (Mutedi & Tjahjono, n.d.).

Metode *vulnerability assessment* dalam penelitian ini digunakan untuk melakukan *scanning* terhadap suatu *website* untuk mendeteksi kerentananan *website* tersebut terhadap *SQL Injection* (Laksono & Santoso, 2021). *SQL Injection* merupakan salah satu teknik peretasan dengan cara menyalahgunakan celah keamanan yang ada di lapisan *SQL* berbasis data pada suatu aplikasi atau *website* (Chegu et al., 2022). Terbentuknya celah tersebut akibat input yang tidak difilter dengan benar dalam pembuatannya, sehingga terciptalah celah yang bisa disalah gunakan.

2. METODE

Penelitian dilakukan dengan menggunakan metode *vulnerability assessment*, dimana penelitian ini berfokus pada tahapan identifikasi *website* dan *vulnerability scanning* untuk melakukan pengujian kerentanan menggunakan *tools vulnerability scanner*, kemudian melakukan analisis terhadap hasil pengujian. Berikut tahapan yang dilakukan dalam penelitian ini.

Gambar 1. Alur Penelitian

2.1. Identifikasi *Website*

Website yang digunakan merupakan *website* dari Program Studi Teknologi Informasi Universitas Muhammadiyah Semarang. Memiliki alamat *website* ti.unimus.ac.id, menggunakan *domain* unimus.ac.id, menggunakan server: denver.ns.cloudflare.com dan luciana.ns.cloudflare.com.

Gambar 2. Halaman website Ti Unimus

2.2. *Penetration testing*

Kegiatan untuk melakukan evaluasi keamanan dari suatu sistem jaringan komputer adalah dengan menggunakan *penetration testing* (McKinnel et al., 2019). Hasil *testing* tersebut akan menghasilkan beberapa kerentanan-kerentanan yang dapat digunakan sebagai celah oleh peretas. *Penetration testing* dapat digunakan untuk meminimalisir terjadinya hal-hal yang tidak diinginkan salah satunya adalah untuk mengidentifikasi serangan ijeksi, terutama injeksi pada database atauserver yang bisa ditemukan dalam kasus *SQL Injection* (Alanda et al., 2021).

2.3. *Vulnerrability Assessment*

Vulnerability Assessment merupakan proses untuk melakukan identifikasi, klasifikasi dan evaluasi terhadap tingkat kerentanan yang terjadi pada suatu sistem teknologi informasi (Alazmi & Leon, 2022). Hasil dari identifikasi *vulnerability assesssment* akan memberikan tingkat kerentanan yang nantinya akan dapat dilakukan evaluasi terhadap kerentanan tersebut (Riadi et al., 2020). Pemeriksaan yang terperinci dan sistematis pada infrastruktur suatu sistem dengan *vulnerability assessment* dapat menentukan kelemahan dan solusi yang bisa dilakukan untuk menanggulangi kerentanan yang ada.

2.4. *Generating Report*

Generating report adalah proses hasil dari data secara otomatis oleh sistem, yang berisikan data dan informasi yang telah dianalisis dan diorganisir sedemikian rupa untuk tujuan tertentu. Proses ini melibatkan pengumpulan data, analisis data, dan penyajian hasil analisis dalam bentuk yang mudah dibaca dan dimengerti.

2.5. OWASP

OWASP adalah sebuah organisasi nirlaba yang fokus pada keamanan *web* aplikasi. OWASP menyediakan beberapa artikel, metodologi, dokumentasi, alat, dan teknologi yang tersedia secara gratis di bidang *IoT*, perangkat lunak sistem, dan keamanan aplikasi web (Firman Ashari et al., n.d.). OWASP menyediakan sumber daya yang gratis dan terbuka sehingga aplikasi OWASP dapat digunakan secara gratis untuk membantu pengguna dalam meningkatkan keamanan aplikasi dan *website* (Wen & Katt, 2023).

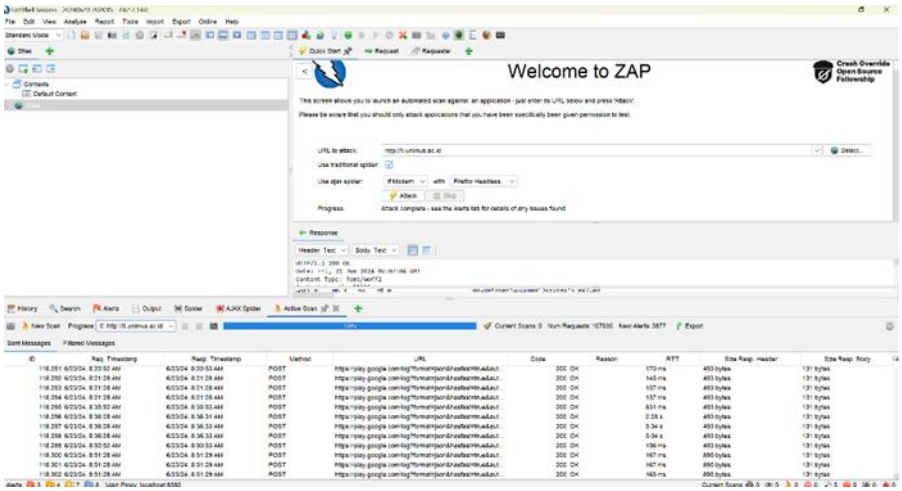
2.6. OWASP ZAP

Zed Attack Proxy (ZAP) adalah aplikasi untuk melakukan scanning untuk menemukan *vulnerabilities* dalam suatu web applications secara mudah, ZAP menyediakan scanner automatis dan manual,menggunakan tool untuk menemukan vulnerabilities secara manual (Elanda & Buana, 2020). Penggunaan aplikasi Zap dalam server memungkinkan pengguna untuk memanipulasi semua lalu lintas yang melewatinya, termasuk lalu lintas menggunakan https dan http. ZAP telah ditambahkan ke dalam Radar Teknologi *ThoughtWorks* pada 30 Mei 2015. Hasil dari kerentanan didalam aplikasi ZAP dapat dianalisis menggunakan OWASP Top 10.

3. HASIL DAN PEMBAHASAN

3.1. *Vulnerability Scanning*

Aplikasi ZAP digunakan untuk melakukan *scanning process*, dengan memasukkan alamat *website* ti.unimus.ac.id. Proses scanning dapat dilihat pada tab *active scan*, hasil dari *active scan* terdapat 107930 *request* dan 3877 *New Alert*s.



Gambar 3. *Scanning* menggunakan *Zap*

Alerts merupakan hasil dari proses *active scan*, *alerts* berisikan informasi tentang kerentanan yang terdapat pada *website* yang telah di*scan*. Terdapat hasil 22 kerentanan yang terbaca pada *tab alerts*. Hasil menunjukan bahwa terdapat 3 kerentanan *High*, 4 *Medium*, 7 *low* dan 8 *Informational*, dengan nilai *confidence* 4 *High*, 11 *Medium* dan 7 *Low*.

Tabel 1. Hasil Vulnerability Scanning

Kerentanan	Confidence		
	High	Medium	Low
High	1	2	
Medium	1	1	2
Low	1	5	1
Informational	1	3	4

Pengujian dilakukan terhadap dua protokol yang berbeda, pada https terdapat 7 kerentanan dan pada protokol http terdapat 14 kerentanan. Https memiliki kerentanan lebih sedikit dikarenakan penggunaan protokol https menggunakan enkripsi berbeda dengan http yang tidak menggunakan enkripsi data.

Tabel 2. Kerentanan protokol *website*

	Kerentanan			
	High	Medium	Low	Informational
https://ti.unimus.ac.id	0	1	3	4
http://ti.unimus.ac.id	3	3	4	4

3.2. Generating Report dan OWASP

Tahap *generating report* yaitu melakukan dokumentasi terkait kerentanan dan hasil analisa kerentanan yang dilakukan, dan memberikan rekomendasi untuk memperbaiki kerentanan pada sistem.

Tabel 3. Hasil *generating report*

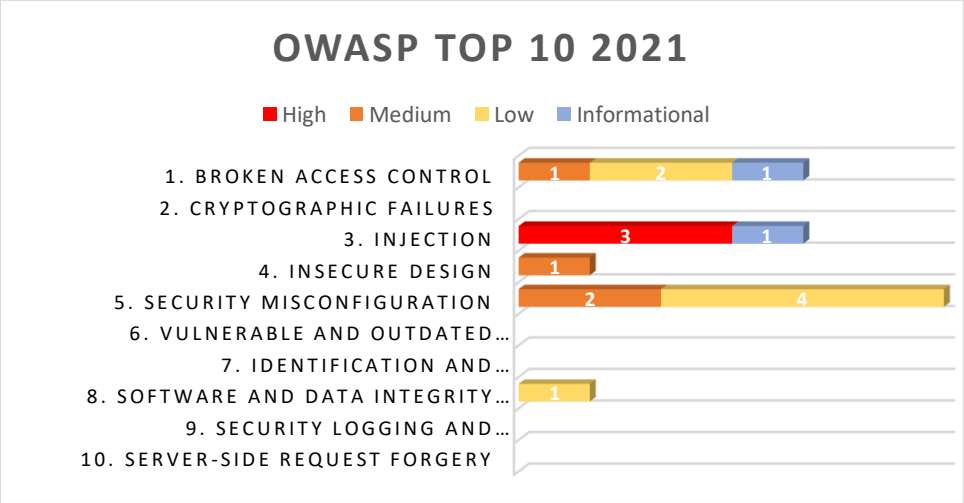
Tipe Alert	Kerentanan	Deskripsi	Pencegahan
SQL Injection - MsSQL	*High*	*SQL injection* bisa terjadi	Jangan melakukan input data yang penting ke dalam server atau database, jika aplikasi menggunakan JDBC, gunakan *Prepared Statement* atau *Callable Statement*, dengan parameter yang telah ditentukan.
SQL Injection - SQLite	*High*	*SQL injection* bisa terjadi	Jangan melakukan input data yang penting ke dalam server atau database, jika aplikasi menggunakan JDBC, gunakan *Prepared Statement* atau *Callable Statement*, dengan parameter yang telah ditentukan.
Server Side Template Injection (Blind)	*High*	Ketika pengguna melakukan input, render data akan dievaluasi oleh mesin template dan kode dapat di eksekusi dari jarak jauh	Bisa mengganti dengan mesin *template* yang aman.
Absence of Anti-CSRF Tokens	*Medium*	Tidak ada token Anti-CSRF yang ditemukan dalam formulir pengiriman HTML. Pemalsuan permintaan lintas situs adalah serangan yang memaksa korban untuk mengirim permintaan HTTP ke tujuan target tanpa sepengetahuan korban. Penyebab utamanya adalah fungsionalitas aplikasi yang menggunakan tindakan URL/formulir yang dapat diprediksi secara berulang. Sifat serangannya adalah CSRF mengeksploitasi kepercayaan yang dimiliki situs web terhadap pengguna.	Fase: Arsitektur dan Desain Gunakan perpustakaan atau kerangka kerja yang terverifikasi yang tidak memungkinkan terjadinya kelemahan ini atau menyediakan konstruksi yang membuat kelemahan ini lebih mudah dihindari. Misalnya, gunakan paket anti-CSRF seperti OWASP CSRFGuard.
Content Security Policy (CSP) Header Not Set	*Medium*	Lapisan keamanan tambahan yang membantu mendeteksi dan memitigasi jenis serangan tertentu,	Pastikan bahwa server web, server aplikasi, *load balancing* dikonfigurasikan secara benar

Tipe Alert	Kerentanan	Deskripsi	Pencegahan
		termasuk *Cross Site Scripting* (XSS) dan serangan injeksi data. Serangan ini digunakan untuk segala hal mulai dari pencurian data hingga perusakan situs atau penyebaran *malware*.	
Missing Anti-clickjacking Header	*Medium*	Respons tidak menyertakan *Content Security-Policy* dengan arahan 'frameancestors' atau *X-Frame-Options* untuk melindungi dari serangan *ClickJacking*	Memastikan salah satu dari HTTP *Content-Security-Policy* dan *XFrame-Options* di konfigurasi di semua halaman web yang dikembalikan oleh situs/aplikasi.
Parameter Tampering	*Medium*	Manipulasi parameter yang menyebabkan halaman kesalahan atau jejak tumpukan Java yang ditampilkan, menunjukkan kurangnya penanganan pengecualian dan potensi area untuk eksploitasi lebih lanjut.	Identifikasi penyebab kesalahan dan perbaiki. Jangan percaya masukan dari sisi klien dan terapkan pemeriksaan ketat di sisi server. Selain itu, temukan pengecualian dengan benar. Gunakan halaman *error* 500 untuk kesalahan server internal.
Application Error Disclosure	*Low*	Halaman yang berisi pesan kesalahan/peringatan yang mungkin mengungkapkan informasi sensitif seperti lokasi file yang menghasilkan pengecualian yang tidak tertangani. Informasi ini dapat digunakan untuk meluncurkan serangan lebih lanjut terhadap aplikasi web.	Meninjau kode sumber halaman. Terapkan halaman khusus untuk kesalahan.Pertimbangkan untuk menerapkan mekanisme untuk memberikan referensi/pengidentifikasi kesalahan unik ke klien (*browser*) serta mencatat detail di sisi server dan tidak memaparkannya kepada pengguna
Cookie without SameSite Attribute	*Low*	*Cookie* telah ditetapkan tanpa atribut *SameSite*, yang berarti bahwa *cookie* dapat dikirim sebagai hasil dari permintaan 'lintas situs'. Atribut *SameSite* merupakan tindakan pencegahan yang efektif terhadap pemalsuan permintaan lintas situs, penyertaan skrip lintas situs, dan serangan pengaturan waktu.	Pastikan atribut situs *SameSite* diatur ke '*lax'* atau idealnya '*strict'* untuk semua *cookie*.

Tipe Alert	Kerentanan	Deskripsi	Pencegahan
Cross-Domain JavaScript Source File Inclusion	*Low*	Laman menyertakan satu atau beberapa file skrip dari domain pihak ketiga.	Pastikan file sumber *JavaScript* dimuat hanya dari sumber terpercaya, dan sumber tidak dapat dikontrol oleh
Secure Pages Include Mixed Content	*Low*	Halaman tersebut memuat konten campuran, yaitu konten yang diakses melalui HTTP, bukan HTTPS.	Halaman website yang tersedia melalui SSL/TLS harus sepenuhnya terdiri dari konten yang dikirimkan melalui SSL/TLS. Halaman tersebut tidak boleh berisi konten apa pun yang dikirimkan melalui HTTP yang tidak terenkripsi. Hal ini termasuk konten dari situs pihak ketiga.
Strict-Transport-Security Header Not Set	*Low*	*HTTP Strict Transport Security* (HSTS) adalah mekanisme kebijakan keamanan web yang dengannya server web menyatakan bahwa agen pengguna yang patuh (seperti peramban web) akan berinteraksi dengannya hanya menggunakan koneksi HTTPS yang aman (yaitu HTTP yang dilapisi TLS/SSL). HSTS adalah protokol jalur standar IETF dan ditetapkan dalam RFC 6797.	Pastikan server web, server aplikasi, menyeimbangkan beban, dikonfigurasi untuk menerapkan *Strict-Transport-Security*.
Timestamp Disclosure - Unix	*Low*	Stempel waktu ditetapkan oleh aplikasi/server web - Unix	Konfirmasikan secara manual bahwa data *timestamp* tidak sensitif, dan bahwa data tidak dapat digabungkan untuk mengungkapkan pola yang dapat dieksploitasi.
X-Content-Type-Options Header Missing	*Low*	*Header Anti-MIME-Sniffing X-Content-Type-Options* tidak ditetapkan ke 'nosniff'. Dapat membuat versi lama Internet Explorer dan Chrome untuk melakukan *MIME-sniffing* pada isi respons, yang berpotensi menyebabkan isi respons ditafsirkan dan ditampilkan sebagai tipe konten selain tipe konten yang dideklarasikan. Versi *Firefox* saat ini (awal 2014)	Pastikan bahwa aplikasi server web menetapkan *header Content-Type* dengan tepat, dan menetapkan *header X-Content-Type-Options* ke '*nosniff*' untuk semua halaman web, pastikan bahwa pengguna akhir menggunakan browser web modern dan sesuai standar yang tidak melakukan *MIME-sniffing* sama sekali, atau yang dapat diarahkan oleh aplikasi

Tipe Alert	Kerentanan	Deskripsi	Pencegahan
		dan versi lama akan menggunakan tipe konten yang dideklarasikan (jika ada yang ditetapkan), daripada melakukan *MIME-sniffing*.	web/server web untuk tidak melakukan *MIME-sniffing*.
Information Disclosure - Suspicious Comments	*Informational*	Respons tersebut tampaknya berisi komentar mencurigakan yang dapat membantu penyerang. Catatan: Pencocokan yang dilakukan dalam blok skrip atau file ditujukan terhadap seluruh konten, bukan hanya komentar.	Hapus semua komentar yang mengembalikan informasi yang dapat membantu penyerang dan perbaiki masalah mendasar sesuai rujukan.
User Controllable HTML Element Attribute (Potential XSS)	*Informational*	Memeriksa masukan yang diberikan pengguna dalam parameter string kueri dan data POST untuk mengidentifikasi di mana nilai atribut HTML tertentu mungkin dikontrol.	Validasi semua masukan dan bersihkan keluaran sebelum menulis ke atribut HTML apa pun.

Hasil dari *alert* dianalisis menggunakan OWASP Top 10 2021, OWASP Top 10 memberikan hasil sebagai berikut:



Gambar 4. Hasil OWASP Top 10

4. KESIMPULAN

Vulnerability scanner menggunakan Zap, memberikan hasil 22 *alerts*, 3 diantaranya memiliki kerentanan yang tinggi (*high risk*). Hasil dari kerentanan *website* ti.unimus.ac.id memiliki *SQL Injenction* yang tinggi, sehingga *website* ti.unimus.ac.id tidak aman digunakan untuk menyimpan data penting. Resiko kerentanan *website* ti.unimus.ac.id dapat dilihat pada *Server Side Template Injection* yang menyatakan bahwa *template website* yang digunakan tidak aman. Solusi untuk meningkatkan

keamanan *website* tersebut adalah dengan melakukan validasi pengguna, instal plugin untuk mencegah *SQL Injection*, membatasi akses pengguna dan mengganti *template website* dengan *template* yang aman

DAFTAR PUSTAKA

Alanda, A., Satria, D., Ardhana, M. I., Dahlan, A. A., & Mooduto, H. A. (2021). Web Application Penetration Testing Using SQL Injection Attack. *JOIV : International Journal on Informatics Visualization*, *5*(3), 320. https://doi.org/10.30630/joiv.5.3.470

Alazmi, S., & Leon, D. C. De. (2022). A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners. *IEEE Access*, *10*, 33200–33219. https://doi.org/10.1109/ACCESS.2022.3161522

Aurangzeb, M., Wang, Y., Iqbal, S., Naveed, A., Ahmed, Z., Alenezi, M., & Shouran, M. (2024). Enhancing cybersecurity in smart grids: Deep black box adversarial attacks and quantum voting ensemble models for blockchain privacy-preserving storage. *Energy Reports*, *11*, 2493–2515. https://doi.org/10.1016/j.egyr.2024.02.010

Chegu, S., Reddy, G. U., Bhambore, B. S., Adeab, K., Honnavalli, P., & Eswaran, S. (2022). An improved filter against injection attacks using regex and machine learning. *Network Security*, *2022*(9). https://doi.org/10.12968/S1353-4858(22)70055-4

Elanda, A., & Buana, R. L. (2020). Analisis Keamanan Sistem Informasi Berbasis Website Dengan Metode Open Web Application Security Project (OWASP) Versi 4: Systematic Review. *CESS (Journal of Computer Engineering, System and Science)*, *5*(2), 185. https://doi.org/10.24114/cess.v5i2.17149

Firman Ashari, I., Rizta Anugrah P, L., Andintya W, N., Denira, S. T., Teknik, J., Fisis, S., Produksi, J. T., Industri, D., Sumatera, T., & Selatan, L. (n.d.). *ANALISIS CELAH KEAMANAN DAN MITIGASI WEBSITE E-LEARNING ITERA MENGGUNAKAN OWASP ZED ATTACK PROXY (ZAP) VULNERABILITY AND MITIGATION ANALYSIS OF THE ITERA E-LEARNING WEBSITE USING OWASP ZED ATTACK PROXY (ZAP)*. http://dinarek.unsoed.ac.id

Fucci, D., Alégroth, E., Felderer, M., & Johannesson, C. (2024). Evaluating software security maturity using OWASP SAMM: Different approaches and stakeholders perceptions. *Journal of Systems and Software*, *214*, 112062. https://doi.org/10.1016/j.jss.2024.112062

Hardiani, T., Wijayanto, D., & Latifah, N. (2022). Data Security Analysis with OWASP Framework on Website XYZ. *CYBERNETICS*, *6*(01), 10–20.

Laksono, A. T., & Santoso, J. D. (2021). Analysis of Website Security of SMKN 1 Pangandaran Against SQL Injection Attack Using OWASP Method. *The IJICS (International Journal of Informatics and Computer Science)*, *5*(2), 209. https://doi.org/10.30865/ijics.v5i2.3208

McKinnel, D. R., Dargahi, T., Dehghantanha, A., & Choo, K.-K. R. (2019). A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment. *Computers & Electrical Engineering*, *75*, 175–188. https://doi.org/10.1016/j.compeleceng.2019.02.022

Mutedi, A., & Tjahjono, B. (n.d.). Systematic Literature Review: Preventing SQL Injection Attacks Using Tools OWASP CSR Web Application Firewall. *Maret*, *7*(1), 151–156. https://doi.org/10.32493/informatika.v7i1.17590

Riadi, I., Yudhana, A., & W, Y. (2020). Analisis Keamanan Website Open Journal System Menggunakan Metode Vulnerability Assessment. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, *7*(4), 853–860. https://doi.org/10.25126/jtiik.2020701928

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

Sharma, D., Mittal, R., Sekhar, R., Shah, P., & Renz, M. (2023). A bibliometric analysis of cyber security and cyber forensics research. *Results in Control and Optimization*, *10*, 100204. https://doi.org/10.1016/j.rico.2023.100204

Wen, S.-F., & Katt, B. (2023). A quantitative security evaluation and analysis model for web applications based on OWASP application security verification standard. *Computers & Security*, *135*, 103532. https://doi.org/10.1016/j.cose.2023.103532