Vol. 7 No. 1 Juni 2026 Hal : 207 - 220 E-ISSN: 2723-1089
P-ISSN: 2776-7779

Analisis Keamanan Website XYZ Melalui Uji Keamanan Hybrid SonarQube dan OWASP ZAP

Aulia Rachmawati

Program Studi Sistem Informasi, Fakultas Sains dan Teknologi, Universitas Jambi

*Coresponding Email: auliarachmawati@unja.ac.id

Abstrak

Keamanan aplikasi web saat ini menghadapi tantangan yang sangat kompleks seiring dengan meningkatnya intensitas serangan siber, khususnya terkait injection flaws dan authentication bypass yang mengeksploitasi celah kerentanan baik pada level kode sumber maupun saat aplikasi sedang dijalankan. Meskipun berbagai metode pengujian keamanan statis dan dinamis telah banyak dikembangkan, implementasinya sering kali dilakukan secara terpisah. Pendekatan yang terfragmentasi tersebut cenderung menghasilkan tingkat false positive yang tinggi dan cakupan deteksi yang tidak menyeluruh, sehingga meninggalkan risiko keamanan yang tidak teridentifikasi secara optimal. Penelitian ini bertujuan untuk merancang serta menguji efektivitas pendekatan Hybrid Security Testing yang mengombinasikan pengujian statis berbasis SonarQube (Static Application Security Testing - SAST) dan pengujian dinamis menggunakan OWASP ZAP (Dynamic Application Security Testing - DAST). Eksperimen dilakukan pada lingkungan staging website XYZ dengan parameter evaluasi yang merujuk pada kategori kerentanan OWASP Top 10 2021 serta akurasi identifikasi celah keamanan. Hasil penelitian menunjukkan bahwa penerapan metode hibrida berhasil meningkatkan cakupan deteksi kerentanan kritis melalui integrasi hasil analisis dari kedua instrumen pengujian. Dalam penelitian ini, SonarQube terbukti efektif dalam mengidentifikasi aspek kualitas kode seperti security code smells dan mendeteksi 238 security hotspots pada struktur kode. Sementara itu, OWASP ZAP berhasil memverifikasi 35 alerts celah runtime nyata, termasuk kerentanan Cross-Site Scripting (XSS) dan authentication flaws. Penelitian ini membuktikan bahwa integrasi hasil pengujian SonarQube dan OWASP ZAP mampu memperkuat jaminan keamanan aplikasi web secara komprehensif tanpa harus bergantung pada satu metode pengujian tunggal, sekaligus memberikan data yang lebih akurat untuk proses remediasi.

Kata Kunci: Keamanan Aplikasi Web, OWASP Top 10, OWASP ZAP, SonarQube, Uji keamanan hibrida

Abstract

Web application security currently faces highly complex challenges alongside the increasing intensity of cyberattacks, particularly concerning injection flaws and authentication bypass that exploit vulnerabilities at both the source code level and during application runtime. Although various static and dynamic security testing methods have been extensively developed, their implementation is often conducted in isolation. Such fragmented approaches tend to result in high false-positive rates and incomplete detection coverage, leaving security risks optimally unidentified. This research aims to design and evaluate the effectiveness of a Hybrid Security Testing approach that combines SonarQube-based static testing (Static Application Security Testing - SAST) and dynamic testing using OWASP ZAP (Dynamic Application Security Testing - DAST). Experiments were conducted on the staging environment of website XYZ, with evaluation parameters referring to the OWASP Top 10 2021

vulnerability categories and the accuracy of security flaw identification. The results demonstrate that the application of the hybrid method successfully increases the detection coverage of critical vulnerabilities through the integration of analysis results from both testing instruments. In this study, SonarQube proved effective in identifying code quality aspects such as security code smells and detecting 238 security hotspots within the code structure. Meanwhile, OWASP ZAP successfully verified 35 real-time runtime alerts, including Cross-Site Scripting (XSS) and authentication flaws. This research proves that the integration of SonarQube and OWASP ZAP testing results is capable of strengthening web application security assurance comprehensively without relying on a single testing method, while simultaneously providing more accurate data for the remediation process.

Keywords: Web Application Security, OWASP Top 10, OWASP ZAP, SAST/DAST, SonarQube, Hybrid security testing.

PENDAHULUAN

Perkembangan teknologi informasi mendorong peningkatan penggunaan aplikasi web di berbagai sektor, namun juga meningkatkan risiko serangan siber seperti SQL Injection, XSS, authentication bypass, dan kesalahan konfigurasi [1]. Hal ini menjadikan keamanan web sebagai aspek krusial untuk mencegah kebocoran data, penurunan reputasi, dan kerugian finansial [2]. Oleh karena itu, pengujian keamanan diperlukan dengan mengacu pada standar OWASP Top 10 2021 guna memastikan keandalan dan perlindungan sistem [3][4].

Berbagai metode digunakan dalam pengujian, di antaranya OWASP ZAP dan SonarQube. OWASP ZAP berfungsi mendeteksi kerentanan pada aplikasi saat berjalan serta memberikan rekomendasi perbaikan [5][6], sedangkan SonarQube menganalisis kualitas dan keamanan kode sumber secara statis, termasuk mendeteksi kerentanan dan kepatuhan terhadap standar pengkodean [7][8]. Beberapa penelitian terkait telah dilakukan, pada penelitian [9] menggunakan metode OWASP ZAP dan SQLMap untuk melakukan uji penetrasi terhadap kerentanan keamanan web. Hasil dari penilitian menunjukkan penggunaan dua metode tersebut dapat mengexploitasi celah keamanan pada web khususnya celah untuk serangan SQL injection. Selanjutnya pada penelitian [10] membandingkan metode OWASP ZAP dengan Arachni untuk menilai kerentanan keamanan web. Hasil penelitian menunjukkan OWASP ZAP lebih unggul dalam melakukan

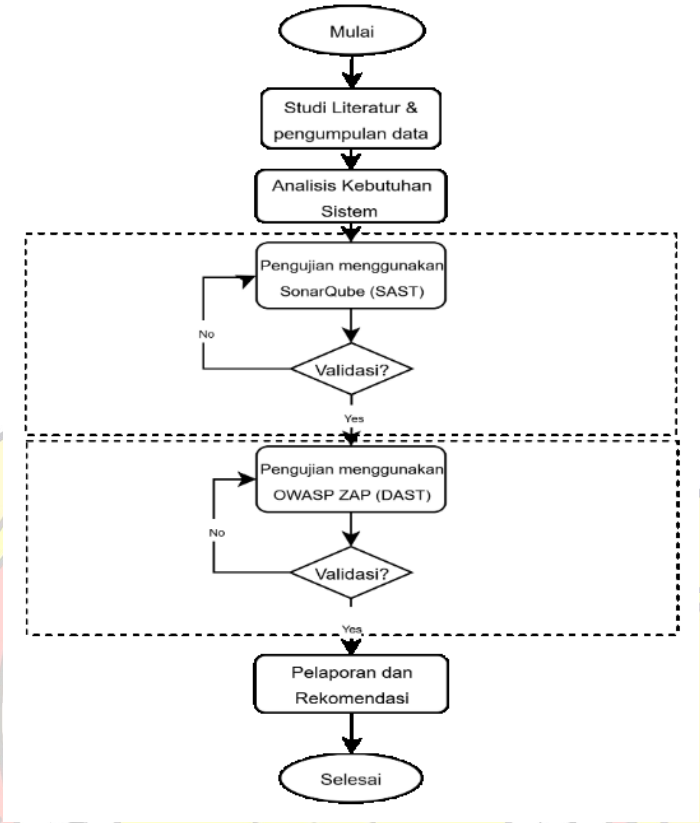
vulnerability assessment. Pada penelitian [11] menggunakan metode OWASP untuk menganalisis kerentanan keamanan web. Hasil dari penelitian ini menunjukkan OWASP ZAP mampu mendeteksi celah keamanan seperti Cross-Site Scripting (XSS), Clickjacking, dan Man-in-the-Middle. Pada penelitian [12] dilakukan analisis penggunaan SonarQube sebagai salah satu tools untuk analisis kode, hasil penelitian menunjukkan bahwa SonarQube memiliki fleksibilitas serta fitur keamanannya yang kuat dan berfokus pada keamanan menjadikannya pilihan yang tepat untuk melindungi sistem perangkat lunak. Selain itu, pada penelitian [13] SonarQube digunakan untuk menganalisis kode website berbasis laravel. Hasil penelitian menemukan bahwa dengan penggunaan metriks yang luas, SonarQube dapat menjadi acuan pada standarisasi pengkodean yang telah diatur sehingga dapat membantu developer untuk meningkatkan kualitas kode program. Pada penelitian [14] menggunakan SonarQube untuk analisis kode sumber aplikasi website rumah kreatif toba. Hasil penelitian menemukan bahwa SonarQube mampu mengidentifikasi titik kerentanan keamanan sehingga berkontribusi pada proses pengembangan yang lebih efisien dan berkelanjutan.

Berdasarkan hal tersebut, penelitian ini menerapkan pendekatan Hybrid Security Testing dengan mengombinasikan SAST (SonarQube) dan DAST (OWASP ZAP). Pendekatan ini dipilih karena SAST mampu menganalisis kode sebelum dijalankan, sementara DAST menguji aplikasi pada kondisi runtime [15][16]. Tujuan penelitian adalah mengidentifikasi kerentanan keamanan, membandingkan kedua metode, serta memberikan rekomendasi peningkatan keamanan aplikasi web XYZ secara komprehensif.

METODE PENELITIAN

Penelitian ini menggunakan kerangka kerja untuk menganalisis kerentanan pada website dengan pendekatan pengujian hybrid yang memanfaatkan OWASP ZAP dan SonarQube. Objek penelitian yang digunakan adalah website XYZ yang diuji

terhadap berbagai potensi serangan keamanan. Adapun metode penelitian yang digunakan ditunjukkan pada Gambar 1.



Gambar 1. Metode Penelitian

Berdasarkan Gambar di atas tahapan penelitian ini diawali dengan Studi literatur dan pengumpulan data, Analisis kebutuhan sistem, pengujian kualitas kode menggunakan SonarQube (SAST), pengujian aplikasi menggunakan OWASP ZAP (DAST), pelaporan dan rekomendasi.

A. Analisis Kebutuhan Sistem

Penelitian ini membutuhkan perangkat keras dan perangkat lunak untuk mendukung proses pengujian secara hybrid serta analisis terhadap kerentanan pada website XYZ. Selain itu, digunakan pula bahan penelitian berupa data, standar, dan pedoman yang menjadi acuan selama proses pengujian agar hasil yang

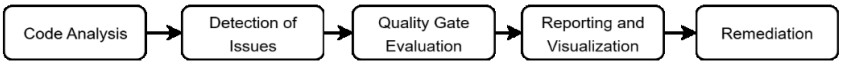
diperoleh akurat dan sesuai dengan tujuan penelitian. Adapun alat dan bahan yang digunakan disajikan pada Tabel 1.

Tabel 1. Analisis Kebutuhan Sistem

No	Alat/Bahan	Spesifikasi	Keterangan
1	Laptop	OS Windows 10 64 bit Processor: AMD Quad Core R5 – 2500U, 3.6 GHz, RAM: 8GB Hard Disk 1TB SSD: 224GB	Alat untuk menjalankan eksperimen
2	SonarQube	Versi 25.9.0.112764	Alat untuk pengujian kode program
3	Sonar Scanner	Versi 8.0.1.6346	
4	OWASP ZAP	Versi 2.17.0	Alat untuk pengujian aplikasi saat berjalan
5	Web Browser	Google Chrome	Alat untuk menjalankan aplikasi sebagai taerget uji
6	Website XYZ	Framework: Code Igniter Bahasa Pemrograman: PHP, CSS, HTML, Javascript	Objek Penelitian
7	OWASP Top 10 2021	-	Standard uji 10 risiko keamanan aplikasi web

B. SonarQube (SAST)

SonarQube merupakan salah satu teknik pengujian yang digunakan untuk menganalisis kode sumber. Pengujian menggunakan SonarQube termasuk dalam teknik pengujian statis karena dilakukan pada saat *pre-runtime*. Tahapan pengujian menggunakan SonarQube ditunjukkan pada Gambar 2.

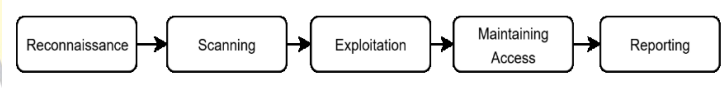


Gambar 2. Tahapan Pengujian Menggunakan SonarQube

1. *Code analysis*: Menganalisis kode sumber website tanpa dieksekusi (*Static Analysis*).
2. *Detection of Issues*: Mengidentifikasi *bug, vulnerabilities,* dan *code smells* berdasarkan standar keamanan OWASP Top 10.
3. *Quality Gate Evaluation*: Menilai apakah kode memenuhi standar keamanan dan kualitas.
4. *Reporting & Visualization*: Menyajikan hasil analisis dalam dashboard interaktif beserta detail dan tingkat keparahan.
5. *Remediation*: Perbaikan kode berdasarkan temuan dan rekomendasi SonarQube.

C. OWASP ZAP (DAST)

OWASP ZAP merupakan salah satu *tools* teknik pengujian yang digunakan untuk menganalisis celah keamanan pada suatu website. Pengujian menggunakan OWASP ZAP termasuk pengujian secara dinamis karena dilakukan dengan teknik *reconnaissance* untuk mengidentifikasi struktur aplikasi, endpoint, serta parameter input dalam keadaan runtime. Dalam melakukan pengujian menggunakan OWASP ZAP mengacu pada tahapan kerangka kerja OWASP ZAP yang ditunjukkan pada Gambar 3.



Gambar 3. Tahapan Pengujian Menggunakan OWAP ZAP

1. *Reconnaissance*: Tahap awal pengumpulan informasi tentang website target.
2. *Scanning*: Pemindaian otomatis untuk menemukan potensi celah keamanan.
3. *Exploitation*: Menguji celah yang ditemukan dengan serangan terkontrol untuk memverifikasi kerentanan.
4. *Maintaining Access*: mempertahankan akses setelah mengeksploitasi celah.
5. *Reporting*: Menyusun laporan detail berisi hasil pemindaian, kerentanan yang ditemukan, tingkat risiko, dan rekomendasi perbaikan.

D. Pelaporan dan Rekomendasi

Laporan hasil pengujian yang mencakup jenis kerentanan, tingkat keparahan, dan dampak potensial. Memberikan rekomendasi perbaikan bagi pengembang website XYZ baik dari sisi kode maupun aplikasi.

HASIL DAN PEMBAHASAN

Pengujian keamanan pada website XYZ menggunakan pendekatan Hybrid Security Testing menggabungkan analisis statis dan dinamis untuk meningkatkan keamanan aplikasi. Proses dilakukan di lingkungan staging agar skenario serangan tidak mengganggu sistem produksi. Berikut penjelasan tiap tahap pengujian.

A. Pengujian Menggunakan SonarQube (SAST)

Pengujian tahap awal dilakukan melalui pendekatan *Static Application Security Testing* (SAST) menggunakan instansi SonarQube terhadap basis kode website XYZ yang mencakup total 226.560 baris kode. Adapun hasil pemindaian menggunakan SonarQube ditunjukkan pada Gambar 4.

	Lines of Code	Security	Reliability	Maintainability	Security Hotspots	Coverage	Duplications
website XYZ	226,560	0	896	24,855	238	0.0%	13.4%

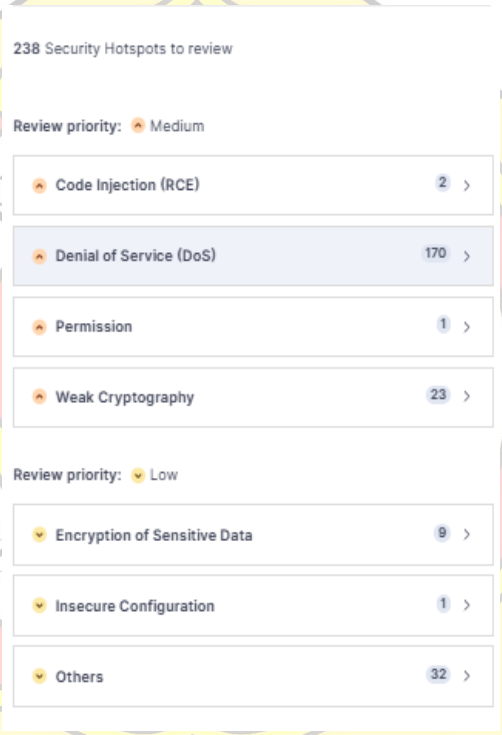
Gambar 4. Hasil Pemindaian Menggunakan SonarQube

Berdasarkan hasil pemindaian pada gambar 4, ditemukan metrik kualitas perangkat lunak sebagai berikut:

1. Analisis Keamanan dan Kerentanan: Hasil pemindaian menunjukkan nilai 0 pada kolom *Security Vulnerabilities*. Hal ini mengindikasikan bahwa secara tekstual, tidak ditemukan pola kode yang merujuk langsung pada celah keamanan terbuka. Namun, ditemukan 238 *Security Hotspots* yang memerlukan peninjauan manual.

2. *Reliability* dan *Maintainability*: Ditemukan sebanyak 896 isu yang berpotensi menyebabkan kegagalan fungsional pada aplikasi. Selain itu, terdapat sekitar 24.855 *code smells* yang secara tidak langsung dapat memperlemah postur keamanan aplikasi dalam jangka panjang.
3. Kualitas Kode dan Duplikasi: Terdapat duplikasi kode sebesar 13,4% dan cakupan pengujian (*coverage*) 0,0%. Tidak adanya unit testing ini menunjukkan tingginya risiko regresi saat dilakukan perbaikan keamanan.

Secara lebih rinci, identifikasi pada bagian *Security Hotspots* ditunjukkan pada Gambar 5.



Gambar 5. Tahapan Pengujian Menggunakan SonarQube

Berdasarkan Gambar 5, risiko didominasi oleh *Denial of Service* (DoS) dengan 170 temuan, diikuti *Weak Cryptography* sebanyak 23 temuan, dan *Code Injection* (RCE) sebanyak 2 temuan pada prioritas menengah. Pada prioritas rendah, terdapat 9 isu enkripsi data sensitif dan 1 isu konfigurasi tidak aman. SonarQube juga mengelompokkan tingkat keparahan menjadi lima kategori, yaitu *blocker, high, medium, low,* dan *info*, dengan hasilnya ditampilkan pada Gambar 6.

Gambar 6. Hasil Pindaian Keparahan Isu pada SonarQube

Berdasarkan Gambar 6 di atas, Tingkat keparahan isu secara keseluruhan didominasi oleh kategori High sebanyak 13.000 temuan dan terdapat 38 isu kategori Blocker yang harus segera ditangani karena dapat menghentikan operasional sistem secara kritis.

B. Korelasi Temuan SonarQube Terhadap Standar OWASP Top 10 2021

Untuk mengevaluasi risiko secara objektif, temuan SAST dipetakan ke standar OWASP Top 10 2021. Hasilnya menunjukkan bahwa kode website XYZ berkaitan dengan beberapa kategori utama, yaitu: **(1) A04:2021 – Insecure Design**, ditandai oleh 170 temuan DoS yang menunjukkan kerentanan pada arsitektur layanan; **(2) A02:2021 – Cryptographic Failures**, terlihat dari 23 isu Weak Cryptography dan 9 isu enkripsi data sensitif yang berpotensi menyebabkan kebocoran data**; (3) A03:2021 – Injection**, dengan 2 temuan Code Injection (RCE) yang berpotensi menjadi celah saat runtime; serta **(4) A05:2021 – Security** Misconfiguration, ditunjukkan oleh 1 isu konfigurasi tidak aman dan sejumlah isu blocker yang mengindikasikan kesalahan konfigurasi mendasar.

C. Rekomendasi Remediasi Berdasarkan Temuan SAST

Berdasarkan temuan Security Hotspots dan isu Blocker yang selaras dengan kategori OWASP Top 10 2021, Adapun rekomendasi tindakan perbaikan (*remediation*) untuk meningkatkan postur keamanan website XYZ dapat dilihat pada tabel 2.

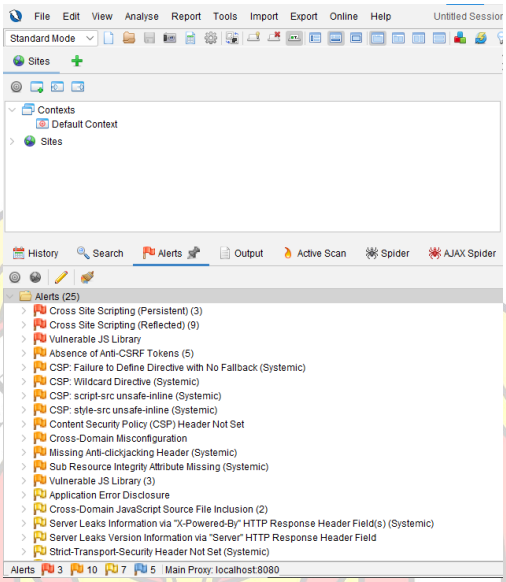
Tabel 2. Rekomendasi Tindakan Perbaikan (Remediation)

Kategori OWASP	Temuan Spesifik	Rekomendasi Tindakan Remediasi
A02:2021 – Cryptographic Failures [3]	Weak Cryptography & Sensitive Data Encryption	Mengganti algoritma hashing usang (seperti MD5/SHA1) dengan Argon2 atau BCrypt, serta memastikan seluruh data sensitif dalam database dienkripsi menggunakan standar AES-256. 1
A03:2021 – Injection [3]	Code Injection (RCE)	Mengimplementasikan validasi input yang ketat (whitelisting) dan menggunakan parameterized queries atau Prepared Statements untuk mencegah eksekusi kode ilegal melalui input pengguna. 2
A04:2021 – Insecure Design [3]	Denial of Service (DoS)	Menerapkan mekanisme Rate Limiting pada API endpoint, melakukan optimasi algoritma yang memakan banyak sumber daya (CPU/RAM), serta mengatur timeout koneksi yang tepat pada sisi server. 3
A05:2021 – Security Misconfiguration [3]	Insecure Configuration & Blocker Issues	Melakukan audit ulang pada file konfigurasi sistem, menonaktifkan fitur/layanan yang tidak diperlukan, serta memastikan atribut cookie menggunakan flag HttpOnly dan Secure. 4

Hasil analisis statis menunjukkan bahwa skor keamanan “0” pada dasbor tidak menjamin aplikasi bebas dari risiko. Banyaknya Security Hotspots dan bug fungsional justru memperluas permukaan serangan. Oleh karena itu, temuan ini menjadi dasar untuk melanjutkan ke tahap DAST guna memvalidasi potensi kerentanan pada kondisi runtime.

C. Pengujian Menggunakan OWASP ZAP (DAST)

Pengujian dinamis dilakukan untuk mengidentifikasi kerentanan pada aplikasi saat sedang dijalankan (runtime). Adapun hasil pindaian menggunakan OWASP ZAP ditunjukkan pada Gambar 7.



Gambar 7. Hasil Pindaian Menggunakan OWASP ZAP

Berdasarkan ringkasan Alerts yang dihasilkan oleh OWASP ZAP, ditemukan total 35 isu keamanan yang terbagi dalam berbagai tingkat risiko: 3 Risiko Tinggi (High), 10 Risiko Sedang (Medium), 17 Risiko Rendah (Low), dan 5 Informasi (Informational).

D. Korelasi Temuan OWASP ZAP Terhadap Standar OWASP Top 10 2021

Temuan ini memberikan bukti empiris adanya celah keamanan yang dapat dieksploitasi dan selaras dengan OWASP Top 10 2021, meliputi: **(1) A03:2021 – Injection (XSS)** dengan temuan Persistent (3) dan Reflected (9) yang berpotensi mencuri sesi dan mengambil alih akun; **(2) A06:2021 – Vulnerable and Outdated Components** melalui penggunaan library JavaScript rentan; **(3) A01:2021 – Broken Access Control** ditandai absennya Anti-CSRF Token (5) dan kesalahan konfigurasi lintas domain; **(4) A05:2021 – Security Misconfiguration** seperti tidak

adanya header keamanan (CSP, HSTS, anti-clickjacking) serta kebocoran informasi header; dan **(5) A04:2021 – Insecure Design** melalui error dan timestamp disclosure.

Hasil DAST ini memverifikasi temuan SAST sebelumnya, khususnya kerentanan XSS akibat validasi input yang lemah, sehingga kombinasi keduanya menghasilkan analisis keamanan yang lebih akurat dan komprehensif.

E. Sintesis dan Perbandingan Hasil Pengujian

Untuk mengevaluasi efektivitas pendekatan hybrid, dilakukan perbandingan antara hasil analisis kode statis dan pengujian dinamis. Adapun perbandingan hasil pengujian menggunakan SonarQube dan OWASP ZAP dapat dilihat pada tabel 3.

Tabel I. Perbandingan Temuan Keamanan: SonarQube VS OWASP ZAP

Kategori Kerentanan (OWASP Top 10)	Temuan SonarQube (SAST)	Temuan OWASP ZAP (DAST)	Status Validasi
A01:2021-Broken Access Control [3]	Tidak Terdeteksi	*Absence of Anti-CSRF Tokens*	Hanya Terdeteksi DAST
A02:2021-Cryptographic Failures [3]	*Weak Cryptography* (23 temuan)	*Insecure Header Configurations*	Terverifikasi Hybrid
A03:2021-Injection (XSS) [3]	*Security Hotspots* (Code Level)	*Persistent & Reflected XSS* (12 temuan)	Terverifikasi Hybrid
A04:2021-Insecure Design [3]	*Denial of Service* (170 temuan)	*Application Error Disclosure*	Terverifikasi Hybrid
A05:2021-Security Misconfiguration [3]	*Insecure Configuration* (1 temuan)	*Missing Security Headers* (HSTS, CSP, dsb)	Terverifikasi Hybrid
A06:2021-Vulnerable Components [3]	*High Maintainability Issues* (25k)	*Vulnerable JS Library*	Terverifikasi Hybrid

Berdasarkan Tabel 3, kedua metode memiliki cakupan deteksi yang berbeda namun saling melengkapi. SonarQube (SAST) unggul dalam mengidentifikasi potensi risiko pada level kode, seperti *Weak Cryptography* dan indikasi *Denial of*

Service. Sementara itu, OWASP ZAP (DAST) mampu membuktikan kerentanan nyata saat runtime, terutama XSS dan ketiadaan Anti-CSRF Token.

Hasil ini menunjukkan bahwa penggunaan satu metode saja masih menyisakan blind spot. Pendekatan hybrid memberikan cakupan deteksi yang lebih luas, mulai dari kode hingga konfigurasi server, sehingga memungkinkan perbaikan yang lebih menyeluruh melalui *refactoring* kode dan penguatan konfigurasi sistem.

SIMPULAN

Penerapan Hybrid Security Testing yang menggabungkan SAST (SonarQube) dan DAST (OWASP ZAP) menunjukkan bahwa kedua metode saling melengkapi melalui validasi lintas pendekatan. Penelitian ini menemukan kerentanan utama sesuai OWASP Top 10 2021, seperti XSS (Injection), Broken Access Control, Cryptographic Failures, dan Insecure Design. Analisis statis mendeteksi 238 security hotspots dan 38 isu kritis pada kode, sedangkan pengujian dinamis mengidentifikasi 35 alert keamanan pada runtime, termasuk Persistent dan Reflected XSS. Pendekatan hibrida ini tidak hanya memperluas cakupan deteksi, tetapi juga meningkatkan akurasi untuk mendukung perbaikan yang lebih menyeluruh, baik melalui refactoring kode maupun penguatan konfigurasi keamanan.

KONFLIK KEPENTINGAN

Penulis menyatakan bahwa tidak terdapat konflik kepentingan

DAFTAR PUSTAKA

[1] D. Budiyanto and M. Mabruri, "Pentingnya Keamanan Siber dalam Era Digital: Tinjauan Global dan Kondisi di Indonesia," *Pros. Semin. Nas. Sains dan Teknol. Seri III Fak. Sains dan Teknol. Univ. Terbuka*, vol. 2, no. 1, pp. 981–994, 2025.

[2] G. Pramuja Inngam Fanani, Muhammad Amirul Mu'min, and N. Tristanti, "Analisis dan Pengujian Kerentanan Website Menggunakan OWASP ZAP," *J. Ris. Sist. dan Teknol. Inf.*, vol. 3, no. 1, pp. 36–50, 2025, doi: 10.30787/restia.v3i1.1886.

[3] I. OWASP Foundation, "OWASP Top Ten | OWASP Foundation," *OWASP Top Ten*, 2023. https://owasp.org/www-project-top-ten/# (accessed May 09, 2026).

[4] F. A. Hassanah, E. Ryansyah, F. M. Setiawan, R. Alamsyah, and A. S. Y. Irawan, "Analisis Kerentanan Keamanan Menggunakan OWASP ZAP dan Pengujian Manual pada Tampilan Antarmuka Laman PDDikti," *JELIKU (Jurnal Elektron. Ilmu Komput. Udayana)*, vol. 13, no. 3, pp. 671–678, 2025, doi: 10.24843/JLK.2025.v13.i03.p14.

[5] A. Fadlil, I. Riadi, and M. A. Mu'min, "Mitigation from SQL Injection Attacks on Web Server Using Open Web Application Security Project Framework," *Int. J. Eng.*, vol. 37, no. 4, pp. 625–634, 2024, doi: 10.5829/ije.2024.37.04a.05.

[6] I. Riadi, A. Fadlil, and M. A. Mu'min, "OWASP Framework-based Network Forensics to Analyze the SQLi Attacks on Web Servers," *MATRIK J. Manajemen, Tek. Inform. dan Rekayasa Komput.*, vol. 22, no. 3, pp. 481–494, 2023, doi: 10.30812/matrik.v22i3.3018.

[7] Sonar, "Code Quality, Security & Static Analysis Tool with SonarQube," 2025. https://www.sonarsource.com/products/sonarqube/ (accessed Jan. 09, 2026).

[8] P. Emanuelsson and U. Nilsson, "A Comparative Study of Industrial Static Analysis Tools," *Electron. Notes Theor. Comput. Sci.*, vol. 217, no. C, pp. 5–21, 2008, doi: 10.1016/j.entcs.2008.06.039.

[9] R. Rahman and D. Fatkhur Razak, "Pengujian Penetrasi Jaringan Menggunakan Owasp Zap Dan Sqlmap Untuk Mengidentifikasi Kerentanan Keamanan Website," *J. Ris. Sist. Inf.*, vol. 1, no. 4, p. 11, 2024.

[10] A. M. Irzan and E. Sulistiyani, "Owasp Zap vs Arachni: Which One is Better in Vulnerability Assesment?," *2024 9th Int. Conf. Informatics Comput. ICIC 2024*, pp. 1–6, 2024, doi: 10.1109/ICIC64337.2024.10956935.

[11] Muhammad Amirul Mu'min, Yana Safitri, Galih Pramuja Inngam Fanani, Setiawan Ardi Wijaya, and Novi Tristanti, "Security Analysis of XYZ Website Using OWASP Zap Tools," *Journix J. Informatics Comput.*, vol. 1, no. 1, pp. 10–20, 2025, doi: 10.63866/journix.v1i1.1.

[12] V. Bhutani, F. G. Toosi, and J. Buckley, "Analysing the Analysers: An Investigation of Source Code Analysis Tools," *Appl. Comput. Syst.*, vol. 29, no. 1, pp. 98–111, 2024, doi: 10.2478/acss-2024-0013.

[13] A. F. Rahmawati and Y. A. Susetyo, "Analisis Quality Code Menggunakan Sonarqube Dalam Suatu Aplikasi Berbasis Laravel," *IT-Explore J. Penerapan Teknol. Inf. dan Komun.*, vol. 2, no. 2, pp. 99–103, 2023, doi: 10.24246/itexplore.v2i2.2023.pp99-103.

[14] N. Sitorus, A. Sirait, E. Lumbantoruan, F. Panjaitan, H. S. H. Harahap, and C. Y. Simangunsong, "Analisis Code Review Menggunakan SonarQube Terhadap Aplikasi Rumah Kreatif Toba Berbasis Website," *J. Comput. Digit. Bus.*, vol. 4, no. 1, pp. 16–23, 2025, doi: 10.56427/jcbd.v4i1.606.

[15] D. Marcilio, C. A. Furia, R. Bonifácio, and G. Pinto, "SpongeBugs: Automatically generating fix suggestions in response to static code analysis warnings," *J. Syst. Softw.*, vol. 168, 2020, doi: 10.1016/j.jss.2020.110671.

[16] F. Awanda Alviansyah and E. Ramadhani, "Implementasi Dynamic Application Security Testing pada Aplikasi Berbasis Android," *Automata*, vol. 2, no. 1, pp. 85–90, 2021.