

Penerapan *DevSecOps* pada *Backend* Sistem Koperasi Karyawan YPTM Universitas Merdeka Malang Menggunakan Pengujian *SAST*

Idham Hastirou ^{1*}, Himawan Pramaditya ²

¹ Program Studi Sistem Informasi, Fakultas Teknologi Informasi, Universitas Merdeka Malang

² Program Studi Sistem Informasi, Fakultas Teknologi Informasi, Universitas Merdeka Malang

Email: ihastiro@gmail.com¹, himawan.pramaditya@unmer.ac.id²

* Penulis Korespondensi: Idham Hastirou

Abstract: Developing a backend system for employee cooperatives requires an approach that not only delivers functional features but also ensures security integration from the earliest stages of the development lifecycle. This research aims to implement DevSecOps on the backend system of YPTM Universitas Merdeka Malang employee cooperative through the integration of Static Application Security Testing (SAST) into the Continuous Integration/Continuous Delivery (CI/CD) pipeline. The backend system is developed using the Bun runtime with oRPC library for type-safe endpoint definitions, PostgreSQL as the relational database, and Redis as the caching layer and asynchronous task queue. DevSecOps implementation is applied across four security layers: development-time, pre-merge, build-time, and runtime, while the evaluation in this study is focused on the pre-merge layer. In the pre-merge layer, SAST scanning using Semgrep runs automatically on every push and pull request event with six rulesets covering OWASP Top 10, security audit, TypeScript, JavaScript, Next.js, and SQL injection. An automated security gate is enforced with thresholds of MAX_CRITICAL=0 and MAX_HIGH=5, which fails the pipeline when vulnerability findings exceed the determined limits. Implementation results show that the pipeline detected 48 security findings, with 42 valid findings and 6 false positives, yielding a precision of 87.5%. The risk score was recorded at 210 based on customized severity weights. All findings are integrated into a risk score trend that is historically recorded for continuous security monitoring. This research demonstrates that SAST integration into the CI/CD pipeline can be implemented automatically on the cooperative backend system, with a security gate mechanism that successfully failed the pipeline in the tested scenarios. As a single case study, the results are descriptive and are not intended to be generalized.

Keywords: DevSecOps; Static Application Security Testing; Semgrep; CI/CD; Backend; Digital Cooperative; Shift-left security.

Abstrak: Pengembangan backend sistem koperasi karyawan memerlukan pendekatan yang tidak hanya menghasilkan fitur fungsional, tetapi juga memastikan aspek keamanan terintegrasi sejak awal siklus pengembangan. Penelitian ini bertujuan untuk menerapkan DevSecOps pada backend sistem koperasi karyawan YPTM Universitas Merdeka Malang melalui integrasi Static Application Security Testing (SAST) ke dalam pipeline Continuous Integration/Continuous Delivery (CI/CD). Backend sistem dikembangkan menggunakan runtime Bun dengan library oRPC sebagai definisi endpoint type-safe, PostgreSQL sebagai basis data relasional, serta Redis sebagai lapisan cache dan antrian tugas asinkron. Penerapan DevSecOps dilakukan melalui empat lapisan keamanan, yaitu development-time, pre-merge, build-time, dan runtime, sedangkan evaluasi pada penelitian ini difokuskan pada lapisan pre-merge. Pada lapisan pre-merge, pemindaian SAST menggunakan Semgrep dijalankan secara otomatis pada setiap event push dan pull request dengan enam ruleset yang mencakup OWASP Top 10, security audit, TypeScript, JavaScript, Next.js, dan SQL injection. Security gate otomatis diterapkan dengan ambang batas MAX_CRITICAL=0 dan MAX_HIGH=5, yang menggagalkan pipeline apabila temuan kerentanan melampaui threshold yang ditentukan. Hasil implementasi menunjukkan bahwa pipeline mendeteksi 48 temuan keamanan, dengan 42 temuan valid dan 6 false positive, menghasilkan precision sebesar 87,5%. Risk score tercatat sebesar 210 berdasarkan bobot severity yang dikustomisasi. Seluruh temuan terintegrasi ke dalam risk score trend yang dicatat secara historis untuk pemantauan tren keamanan berkelanjutan. Penelitian ini menunjukkan bahwa integrasi SAST ke dalam pipeline CI/CD dapat diterapkan secara otomatis pada backend sistem koperasi, dengan mekanisme security gate yang

Diterima: 14 Agustus 2026

Direvisi: 27 Agustus 2026

Disetujui: 22 September 2026

Tersedia: 30 September

Diterbitkan: 30 September 2026



Hak cipta: © 2026 oleh penulis.

Diserahkan untuk kemungkinan

publikasi akses terbuka

berdasarkan syarat dan ketentuan

lisensi Creative Commons

Attribution (CC BY SA) (

<https://creativecommons.org/licenses/by-sa/4.0/>)

berhasil menggagalkan pipeline pada skenario pengujian yang dilakukan. Sebagai studi kasus tunggal, hasil ini bersifat deskriptif dan tidak dimaksudkan untuk digeneralisasi.

Kata kunci: DevSecOps; Static Application Security Testing; Semgrep; CI/CD; Backend; Koperasi Digital; Shift-left security.

1. Pendahuluan

Dewasa ini perkembangan teknologi dan informasi telah mengalami perkembangan pesat, yang telah mendorong berbagai organisasi untuk meningkatkan efisiensi operasional melalui pemanfaatan sistem berbasis teknologi. Koperasi sebagai lembaga ekonomi berbasis keanggotaan juga dituntut untuk berpacu dalam meningkatkan kualitas layanan kepada anggota. Namun, pada praktiknya masih banyak koperasi yang belum memanfaatkan teknologi sebagai proses operasional seperti pengelolaan anggota, transaksi, dan layanan simpan pinjam. Kondisi ini menyebabkan keterbatasan dalam efisiensi kerja, transparansi data, serta kecepatan layanan kepada anggota [1].

Koperasi Karyawan YPTM Universitas Merdeka Malang (Kopkar UNMER) merupakan koperasi yang melayani kebutuhan transaksi dan layanan keuangan bagi anggota UNMER Malang. Seiring meningkatnya kebutuhan digitalisasi layanan, koperasi membutuhkan sistem yang mampu mengintegrasikan berbagai layanan bisnis seperti transaksi, simpanan, pinjaman, *voucher*, dan kredit secara terpusat. Oleh karena itu, dikembangkan sistem koperasi digital bernama KopkarPay sebagai upaya meningkatkan kualitas layanan dengan fitur yang terintegrasi pada koperasi berbasis teknologi digital.

Digitalisasi koperasi terbukti mampu memudahkan operasional bisnis, meningkatkan akses layanan bagi anggota, serta menghubungkan pelaku usaha dengan pasar secara lebih luas [1]. Sistem informasi koperasi berbasis komputer juga meningkatkan efektivitas pengelolaan anggota dan mempercepat proses transaksi [2]. Transformasi digital ini tidak hanya menuntut peningkatan fitur layanan, tetapi juga aspek keamanan sistem yang lebih baik. Pengelolaan data sensitif seperti data anggota, transaksi keuangan, dan pengajuan pinjaman rentan terhadap ancaman keamanan apabila tidak dirancang dengan pendekatan yang tepat.

Aspek keamanan dan transparansi data diatur dalam regulasi pemerintah. Berdasarkan Undang-Undang Nomor 27 Tahun 2022 tentang Perlindungan Data Pribadi, setiap penyelenggara sistem elektronik wajib menjamin keamanan data pribadi [3]. Tanpa mekanisme keamanan yang baik sejak tahap pengembangan, risiko kebocoran data, manipulasi transaksi, serta gangguan layanan dapat menurunkan kepercayaan anggota terhadap koperasi.

Penelitian terdahulu menunjukkan bahwa sebagian besar pengembangan sistem koperasi masih berfokus pada fungsionalitas tanpa mengintegrasikan aspek keamanan secara komprehensif [4]. Penelitian yang membahas penerapan DevSecOps masih terbatas [5], dan implementasi SAST dalam konteks sistem koperasi digital belum banyak diterapkan. Pemilihan SAST pada penelitian ini didasarkan pada kemampuannya mendeteksi kerentanan pada kode sumber sebelum aplikasi dijalankan, mendukung konsep *shift-left security*, serta kemudahan integrasinya ke dalam *pipeline* CI/CD [6][7]. Dibandingkan *Dynamic Application Security Testing* (DAST) yang memerlukan aplikasi berjalan di lingkungan *runtime*, dan *Interactive Application Security Testing* (IAST) yang membutuhkan sensor khusus pada *runtime* aplikasi [8], SAST lebih ringan dan memberikan umpan balik lebih awal kepada tim pengembang.

Masalah teknis yang belum terjawab pada studi terdahulu adalah bagaimana pemindaian SAST dapat diotomatisasi pada *pipeline* CI/CD backend koperasi, lengkap dengan kebijakan penghentian *pipeline* (*security gate*) dan pemantauan risiko dari waktu ke waktu. Untuk itu, penelitian ini menjawab tiga pertanyaan penelitian (*research question*, RQ): RQ1, bagaimana SAST dapat diintegrasikan ke dalam *pipeline* CI/CD pada backend KopkarPay? RQ2, berapa *precision* hasil pemindaian Semgrep pada pengamatan yang dilakukan? RQ3, apakah *security gate* berbasis *threshold* dan pemetaan *severity* mampu menggagalkan *pipeline* ketika temuan melampaui batas yang ditetapkan?

Kontribusi penelitian ini adalah rancangan integrasi *security gate* berbasis pemetaan *severity* yang dapat dikustomisasi (*severity-map*, *rule_overrides*, dan *false_positive_exceptions*) serta pencatatan tren *risk score* historis pada backend koperasi

digital. Rancangan ini dapat direplikasi pada proyek lain dengan stack serupa. Penelitian ini merupakan studi kasus tunggal, sehingga kontribusinya bersifat praktis dan deskriptif.

2. Tinjauan Literatur

Penerapan digitalisasi pada koperasi berdampak signifikan pada efisiensi operasional, transparansi, serta akurasi pengelolaan data [1][2]. Pengembangan *backend* modern kini banyak memanfaatkan ekosistem berbasis RESTful API menggunakan *runtime* berkinerja tinggi seperti Bun untuk menjamin fungsionalitas dan skalabilitas [9]. Integrasi pendekatan DevOps juga mulai diadopsi dalam rancangan kebutuhan sistem koperasi digital guna mempercepat siklus pengiriman fitur [10].

DevSecOps merupakan evolusi dari DevOps yang secara eksplisit mengintegrasikan aspek keamanan ke seluruh tahapan *Software Development Life Cycle* (SDLC) [5]. Myrbakken dan Colomo-Palacios mendefinisikan DevSecOps sebagai pendekatan yang menekankan otomatisasi pengujian keamanan dan kolaborasi antara tim pengembang, operasional, dan keamanan [7]. Konsep inti yang mendasarinya adalah *shift-left security*, yaitu memindahkan aktivitas pengujian keamanan ke tahap paling awal pengembangan sehingga biaya perbaikan kerentanan dapat ditekan secara signifikan, Rajapakse et al. mengidentifikasi tantangan utama adopsi DevSecOps di antaranya adalah kurangnya alat otomatisasi yang terintegrasi dan resistensi budaya tim pengembang terhadap perubahan proses [6].

Zhao et al. menyimpulkan melalui multi-vocal literature review bahwa terdapat lima dimensi utama DevSecOps yaitu definisi, tantangan, praktik, tools, dan metrik [5]. Studi ini menegaskan bahwa alat seperti SAST, DAST, dan SCA merupakan komponen esensial dalam ekosistem DevSecOps modern.

SAST bekerja dengan menganalisis kode sumber secara statis tanpa menjalankan aplikasi, memungkinkan deteksi kerentanan seperti SQL Injection, hardcoded credentials, dan insecure input validation sejak fase penulisan kode [8]. OWASP *Top Ten* 2021 menetapkan sepuluh kategori risiko keamanan aplikasi web yang paling kritis, meliputi Broken Access Control, Cryptographic Failures, dan *Injection*, yang sebagian di antaranya dapat dideteksi melalui pemindaian SAST, bergantung pada jenis kerentanan, rule, dan konfigurasi yang digunakan [11]. Koneru melaporkan bahwa integrasi SAST dalam *pipeline* CI/CD dapat menurunkan kerentanan di lingkungan produksi hingga 30–50% [8]. Shahin et al. menegaskan bahwa continuous integration dan delivery merupakan fondasi penting untuk menerapkan pengujian otomatis secara konsisten pada setiap iterasi pengembangan [12].

Sebagian besar penelitian koperasi masih berfokus pada pengembangan fitur dan pengelolaan hak akses pengguna tanpa mengintegrasikan keamanan secara menyeluruh dalam siklus pengembangan [4]. Kekosongan ini menjadi dasar penelitian yang mengintegrasikan pendekatan DevSecOps dengan SAST pada *backend* sistem koperasi digital KopkarPay.

3. Metode

Metodologi penelitian ini menggunakan pendekatan rekayasa perangkat lunak dengan metode *Research and Development* (R&D) yang berfokus pada otomatisasi keamanan *backend* KopkarPay. Arsitektur keamanan diintegrasikan ke dalam empat lapis utama: *development-time*, *pre-merge*, *build-time*, dan *runtime* [13]. Evaluasi pada penelitian ini difokuskan pada lapisan *pre-merge*.

Backend dikembangkan menggunakan *runtime* Bun dengan *library* oRPC sebagai definisi endpoint *type-safe*, PostgreSQL sebagai basis data relasional, dan Redis sebagai lapisan *cache* dan antrian tugas asinkron. Seluruh kode dikelola dalam arsitektur *monorepo* menggunakan Bun Workspace, yang memisahkan aplikasi (apps/) dan kode bersama (packages/) untuk mendorong modularitas.

Fokus utama otomatisasi ditekankan pada fase *pre-merge* menggunakan platform GitHub Actions. Pemindaian kode sumber dijalankan secara otomatis memanfaatkan perkakas Semgrep via Docker dengan menerapkan enam *ruleset* standar industri, yaitu OWASP *Top 10*, *security-audit*, TypeScript, JavaScript, Next.js, dan SQL *injection* [11]. Pada event *pull request*, flag *--baseline-commit* diaktifkan secara kondisional sehingga pemindaian hanya mencakup file yang berubah (*diff-only scan*), bukan seluruh *codebase*.

Untuk memastikan kode yang rentan tidak masuk ke lingkungan produksi, penelitian ini menerapkan mekanisme *Security gate* berbasis kode melalui script *gate.ts*. Mekanisme ini membaca hasil *semgrep.json*, memetakan setiap temuan ke tier severity menggunakan *severity-*

map.json yang dikustomisasi, menghitung *risk score* terbobot dengan formula $Risk\ score = (Critical \times 10) + (High \times 8) + (Medium \times 5) + (Low \times 2)$, kemudian membandingkan dengan ambang batas $MAX_CRITICAL=0$ dan $MAX_HIGH=5$. *Pipeline* dinyatakan *FAIL* dan dihentikan (*exit code 1*) apabila temuan melampaui *threshold* tersebut. Seluruh laporan diunggah sebagai *GitHub Actions Artifact* (retensi 90 hari) dan metrik historis dicatat otomatis ke *risk-score-trend.json* pada setiap *push* ke *branch main*.

Dasar *threshold* dan bobot. Nilai $MAX_CRITICAL=0$ dan $MAX_HIGH=5$ serta bobot 10/8/5/2 merupakan keputusan desain penulis, bukan hasil penurunan empiris. *Threshold Critical* ditetapkan nol karena sistem mengelola data anggota dan transaksi keuangan, sedangkan batas lima temuan *High* diberikan sebagai toleransi agar *pipeline* tidak terus terblokir pada tahap awal adopsi. Bobot mengikuti urutan tingkat bahaya, yaitu semakin tinggi *severity* semakin besar bobotnya. Keputusan *gate* hanya ditentukan oleh $MAX_CRITICAL$ dan MAX_HIGH ; *risk score* digunakan sebagai metrik pemantauan tren, bukan sebagai kriteria gagal.

Prosedur validasi temuan. Setiap temuan diklasifikasikan sebagai valid atau *false positive* secara manual oleh penulis. Temuan dinyatakan *false positive* apabila bukan merupakan kerentanan sebenarnya pada konteks penggunaannya, misalnya kredensial tiruan yang sengaja dipakai pada lingkungan CI untuk keperluan

type-checking. *Precision* dihitung sebagai jumlah temuan valid dibagi total temuan. Metrik lain seperti *recall* dan *alert fatigue* tidak diukur pada penelitian ini.

Periode dan unit pengamatan. Hasil dicatat pada tiga titik pencatatan *risk-score-trend*, yaitu 23 Juni 2026 (baseline), 30 Juni 2026, dan 1 Juli 2026, pada satu repositori monorepo KopkarPay.

4. Hasil dan Pembahasan

Sistem otomatisasi berhasil diimplementasikan dan diuji pada repositori *monorepo* KopkarPay. *Pipeline* CI/CD berjalan konsisten pada tujuh *workflow* GitHub Actions yang terbagi dalam dua kategori: CI *Workflows* (*ci-security*, *ci-server*, *ci-backoffice*, *ci-web*) sebagai *merge gate*, dan *Build Workflows* (*build-server*, *build-backoffice*, *build-web*) sebagai *deploy gate*.

Berdasarkan riwayat eksekusi *pipeline*, pemindaian SAST menghasilkan data metrik sebagai berikut:

Tabel 1. Hasil Pemindaian Keamanan Semgrep

Parameter Evaluasi	Hasil Pemindaian
Temuan Valid	42 temuan
<i>False positive</i>	6 temuan
Total Temuan	48 temuan
<i>Precision</i>	87,5%
<i>Risk score</i>	210

Precision mencapai 87,5% (42 dari 48 temuan dinilai valid). Enam *false positive* yang teridentifikasi berasal dari kredensial palsu yang sengaja digunakan dalam lingkungan CI untuk keperluan *type-checking*, bukan kredensial sesungguhnya. Temuan ini dikecualikan secara eksplisit melalui mekanisme *false_positive_exceptions* pada *severity-map.json*, sehingga *gate* tetap fokus pada kerentanan yang relevan. *Precision* hanya menggambarkan proporsi temuan valid dari seluruh *alert*; nilai ini belum cukup untuk menyimpulkan tingkat *alert fatigue* karena jumlah *alert* per PR dan waktu *triage* tidak diukur.

Tabel 2. Tren Historis *Risk Score Pipeline* SAST.

Periode	Total Temuan	Valid	FP	Precision	<i>Risk score</i>
---------	--------------	-------	----	-----------	-------------------

23 Juni 2026 (<i>Baseline</i>)	4	4	0	100%	20
30 Juni 2026	4	4	0	100%	20
1 Juli 2026	48	42	6	87,5%	210

Data pada Tabel 2 menunjukkan pemindaian mencatat hasil yang sama pada *baseline* hingga 30 Juni 2026 dengan *risk score* 20 dan precision 100%. Lonjakan pada 1 Juli 2026 menjadi 48 temuan dipicu oleh penambahan kode baru yang memperluas cakupan pemindaian, termasuk file *workflow* GitHub Actions. Sebagian besar temuan berasal dari rule mutable-action-tag yang mendeteksi 44 penggunaan version tag (@v4) yang belum menggunakan *pinned* SHA. Temuan ini sejalan dengan tren penelitian yang mulai memperlakukan GitHub Actions sebagai *dependency* pihak ketiga yang turut membawa risiko keamanan, sebagaimana *dependency* kode pada umumnya [14].

Pada skenario pengujian, mekanisme *security gate* berfungsi sebagaimana dirancang. Ketika *rule_overrides* diaktifkan untuk rule *yaml.github-actions.security.workflow-run-target-code-check-out* yang mendeteksi pola pwn-request (CWE-913) keempat temuan yang sebelumnya tercatat sebagai Medium ter-elevate ke *Critical*, menyebabkan gate mengeluarkan status *FAIL* dan menghentikan seluruh job yang bergantung padanya (*lint*, *typecheck*, *test*, *migrate-dryrun*). Hal ini menunjukkan bahwa konfigurasi *severity_map.json* yang dikustomisasi dapat menegakkan kebijakan keamanan yang lebih ketat terhadap pola kerentanan spesifik yang tidak diklasifikasikan tinggi oleh Semgrep secara bawaan. Pengamatan ini berasal dari satu skenario pengujian dan belum mencakup rekaman gate *PASS/FAIL* dalam jumlah besar.

Secara konseptual, pengujian otomatis pada setiap push/PR memberi umpan balik yang lebih cepat dan hasil yang lebih konsisten dibandingkan pengujian keamanan manual yang bersifat periodik [8][12]. Perbandingan kuantitatif dengan pengujian manual tidak dilakukan pada penelitian ini. Hasil ini sejalan dengan arah temuan Koneru yang melaporkan manfaat integrasi SAST dalam CI/CD [8], namun penelitian ini tidak mengukur penurunan kerentanan di lingkungan produksi.

Keterbatasan. Evaluasi bersifat studi kasus tunggal pada satu backend dengan tiga titik pencatatan. Rincian distribusi severity per pemindaian dan jumlah eksekusi *pipeline* tidak disajikan secara lengkap karena konfigurasi SAST pada repositori telah dinonaktifkan setelah periode pengamatan. Oleh karena itu, hasil dimaknai sebagai gambaran deskriptif implementasi, bukan generalisasi efektivitas SAST. Keterbatasan implementasi lainnya meliputi: tidak digunakannya GitHub Advanced Security (GHAS) sehingga laporan SARIF belum dimanfaatkan secara aktif; tidak adanya DAST maupun IAST untuk mendeteksi kerentanan pada saat runtime, padahal studi komparatif menunjukkan SAST tetap unggul dalam precision (0,98) dibanding DAST (0,23) maupun IAST (0,4), meski masing-masing pendekatan menemukan kerentanan unik yang tidak terdeteksi oleh pendekatan lain [15]; belum ada Software Composition Analysis (SCA) untuk *dependency* pihak ketiga; cakupan *secret Scanning* yang masih terbatas pada CWE-798 melalui Semgrep tanpa dedicated tool seperti Gitleaks; serta belum adanya visualisasi tren otomatis dari data historis *risk-score-trend.json* [6][7].

5. Kesimpulan

Penelitian ini berhasil mengimplementasikan pendekatan *DevSecOps* pada *backend* koperasi digital KopkarPay melalui otomatisasi pengujian SAST menggunakan Semgrep yang terintegrasi dalam *pipeline* GitHub Actions. Hasil pengujian menunjukkan *pipeline* CI/CD mampu mendeteksi kerentanan kode secara dini dengan tingkat precision 87,5%. Penerapan *security gate* otomatis berbasis *threshold* berhasil menggagalkan *pipeline* pada skenario pengujian yang dilakukan. Mekanisme *rule_overrides* pada *severity-map.json* terbukti mampu meningkatkan sensitivitas deteksi terhadap pola kerentanan spesifik yang tidak diklasifikasikan tinggi oleh tool secara bawaan. Karena objek penelitian hanya satu backend koperasi, hasil ini tidak dapat digeneralisasi.

Kontribusi Penulis: Konseptualisasi: Idham Hastirou dan Himawan Pramaditya; Metodologi: Idham Hastirou; Perangkat Lunak: Idham Hastirou; Validasi: Idham Hastirou dan Himawan Pramaditya; Analisis formal: Idham Hastirou; Penulisan—persiapan draf asli:

Idham Hastirou; Penulisan—peninjauan dan penyuntingan: Himawan Pramaditya; Supervisi: Himawan Pramaditya.

Pendanaan: Penelitian ini tidak menerima pendanaan eksternal.

Ucapan Terima Kasih: Penulis mengucapkan terima kasih yang sebesar-besarnya kepada Program Studi Sistem Informasi Fakultas Teknologi Informasi Universitas Merdeka Malang, serta Koperasi Karyawan YPTM Universitas Merdeka Malang atas dukungan penyediaan data dan objek penelitian.

Konflik Kepentingan: Penulis menyatakan tidak ada konflik kepentingan.

Referensi

- [1] D. Kamiilah Afrida, E. Wahyu Puji Lestari, F. Lailiya, and A. Fawaiq Suwanan, “Peran Digitalisasi Koperasi Sebagai Pendongkrak UMKM Dalam Pengembangan Ekonomi Wilayah Kota Surabaya,” 2021.
- [2] M. T. Chulkamdi, Z. Wulansari, A. Haryoko, and R. Alfiyanti, “(elektronik) Fakultas Teknik Universitas Islam Balitar,” 2025.
- [3] Republik Indonesia, “Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi,”
- [4] R. Setiawan and A. Faruq, “SISTEM INFORMASI KOPERASI TABURPUJA BERBASIS TANGGUNG RENTENG,” 2019.
- [5] X. Zhao, T. Clear, and R. Lal, “Identifying the primary dimensions of DevSecOps: A multi-vocal literature review ☆,” *J. Syst. Softw.*, vol. 214, p. 112063, 2024, doi: 10.1016/j.jss.2024.112063.
- [6] R. N. Rajapakse, M. Zahedi, M. A. Babar, and H. Shen, “Challenges and solutions when adopting DevSecOps: A systematic review,” *Inf. Softw. Technol.*, vol. 141, p. 106700, Jan. 2022, doi: 10.1016/J.INFSOF.2021.106700.
- [7] R. Myrbakken Håvard and Colomo-Palacios, “DevSecOps: A Multivocal Literature Review,” in *Software Process Improvement and Capability Determination*, A. and O. R. V. and R. T. and D. A. Mas Antonia and Mesquida, Ed., Cham: Springer International Publishing, 2017, pp. 17–29.
- [8] Naga Murali Krishna Koneru, “Integrating Security into CI/CD Pipelines: A DevSecOps Approach with SAST, DAST, and SCA Tools,” *International Journal of Science and Research Archive*, vol. 3, no. 1, pp. 250–265, Oct. 2021, doi: 10.30574/ijrsra.2021.3.1.0080.
- [9] Agil Maulana Nanda Riady, Paniran Paniran, and I Made Budi Suksmadana, “Perancangan Backend Api Berbasis Rest-API pada Aplikasi Rekomendasi Resep Makanan,” *Mars : Jurnal Teknik Mesin, Industri, Elektro Dan Ilmu Komputer*, vol. 2, no. 3, pp. 94–106, Jun. 2024, doi: 10.61132/mars.v2i3.137.
- [10] Ananta Wira Satriaji, “ANALISIS SPESIFIKASI KEBUTUHAN PERANGKAT LUNAK (SKPL) DALAM PENGEMBANGAN APLIKASI KOPERASI DIGITAL BERBASIS MOBILE MENGGUNAKAN METODE DEVOPS (Studi Kasus Pada Koperasi Karyawan,” 2026.
- [11] OWASP Foundation, “OWASP Top 10:2021,” Open Worldwide Application Security Project (OWASP).
- [12] M. Shahin, M. Ali Babar, and L. Zhu, “Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices,” *IEEE Access*, vol. 5, pp. 3909–3943, 2017, doi: 10.1109/ACCESS.2017.2685629.
- [13] A. Perdana, S. Dewi, N. A. Farhana, and D. Febrian, “Comparative Analysis of SDLC and R&D Methods in System Development: A Case Study of Integrity Zone Management System,” *sinkron*, vol. 9, no. 4, pp. 3197–3209, Oct. 2025, doi: 10.33395/sinkron.v9i4.15337.
- [14] L. Williams *et al.*, “Research Directions in Software Supply Chain Security,” *ACM Trans. Softw. Eng. Methodol.*, vol. 34, no. 5, May 2025, doi: 10.1145/3714464.

-
- [15] A. Seth, S. Bhattacharya, S. Elder, N. Zahan, and L. Williams, “Comparing effectiveness and efficiency of Interactive Application Security Testing (IAST) and Runtime Application Self-Protection (RASP) tools in a large java-based system,” *Empir. Softw. Eng.*, vol. 30, no. 3, p. 67, 2025, doi: 10.1007/s10664-025-10621-5.