BAB II
TINJAUAN PUSTAKA

A. Penelitian Terdahulu

Penelitian terdahulu telah dilakukan oleh (Yunanri, 2023) dengan mengambil judul Analisis Keamanan Website Sistem Informasi Administrasi Kependudukan Menggunakan Metode Vulnerability Assesment. Masalah dari penelitian ini yaitu website lembaga kementrian dalam negeri di bidang kependudukan dan pencatatan sipil menyajikan informasi data-data pribadi masyarakat, sehingga sangat penting untuk di jaga kerahasiannya dan tidak boleh di akses oleh orang yang tidak bertanggung jawab. Dalam penelitian ini metode yang digunakan adalah vulnerability assesment, dan menggunakan tools kali linux dan OWASP ZAP. Hasil dari penelitian ini menemukan bahwa website sistem informasi administrasi kependudukan memiliki beberapa peringatan keamanan dengan risiko tinggi.

(Mulyanto et al., 2022) telah melakukan penelitian dengan mengambil judul Analisis Keamanan Website SMA Negeri 2 Sumbawa Besar Menggunakan Metode Penetration Testing (Pentest). Diambilnya penelitian ini di karenakan website sekolah SMA Negeri 2 Sumbawa Besar telah berhasil di retas oleh penjahat informasi yang mengambil data-data penting milik sekolah dan juga mengakibatkan tampilan website berubah. Adapun tahapan dari penelitian ini yaitu di mulai dari Footprinting, Scanning Fingerprinting, Exploit dan Reporting. Hasil dari penelitian ini menemukan beberapa celah keamanan pada website SMA Negeri 2 Sumbawa Besar yang terdeteksi 13 sub file vulnerability dengan status low dan medium.

Penelitian terdahulu telah dilakukan oleh (Wicaksono et al., 2020) dengan mengambil judul Pengujian Celah Keamanan Aplikasi Berbasis Web Menggunakan Teknik Penetration Testing dan Dynamic Application Security Testing (DAST). Penelitian ini menggunakan metode Penetration Testing dan Dynamic Application Security Testing, dan celah yang diuji antara lain seperti Broken Access Control, dan Sql Injection. Hasil penelitian menunjukkan bahwa celah keamanan Broken Access Control dapat dicegah dengan

membuat kode (id) yang sulit untuk ditebak, dan Sql Injection dapat dicegah dengan menggunakan enscape () pada saat pencarian basisdata.

Penerapan Dynamic Application Security Testing (DAST) untuk mendeteksi kerentanan pada aplikasi Android menggunakan Mobile Security Framework (MobSF) sebagai alat pengujian pernah dilakukan oleh (Alviansyah & Ramadhani, 2021). Masalah yang dihadapi dari penelitian ini yaitu banyaknya aplikasi Android yang tidak memenuhi standar keamanan (ISO/IEC 27001), membuatnya rentan terhadap serangan, seperti Sql injection, XSS, dan IDOR. Penelitian ini menggunakan MobSF untuk menguji kerentanan aplikasi melalui pendekatan black box. Penelitian ini menunjukan MobSF secara efektif mendeteksi berbagai celah keamanan pada aplikasi dengan metode pengujian secara dinamis.

Penelitian yang dilakukan oleh (Ariyadi et al., 2023) dengan judul Analisis Kerentanan Keamanan Sistem Informasi Akademik Universitas Bina Darma Menggunakan OWASP. Masalah dari penelitian ini karena data akademik yang dikelola bersifat krusial, menjaga keamanan informasi menjadi prioritas utama. Ancaman terhadap keamanan data, seperti serangan siber, berpotensi menimbulkan dampak negatif yang signifikan, sehingga diperlukan analisis untuk mengidentifikasi dan mengurangi risiko kerentanan. Penelitian ini menggunakan metode Action Research, yang berfokus pada hasil pemindaian kerentanan keamanan website. Pendekatan ini didukung oleh penerapan kerangka kerja OWASP untuk menguji keamanan aplikasi web secara menyeluruh. Hasil penelitian mengungkapkan informasi mengenai port-port yang terbuka, yang berpotensi menjadi titik masuk serangan. Selain itu, solusi untuk meningkatkan keamanan aplikasi web juga disajikan, termasuk langkah-langkah untuk meminimalkan serangan yang mungkin dilakukan oleh peretas.

(Zirwan, 2022) telah melakukan penelitian dengan mengambil judul pengujian dan analisis keamanan website menggunakan Acunetix Vulnerability Scanner. Masalah dari penelitian ini yaitu Institut Teknologi Padang (ITP) harus dapat menyesuaikan diri dengan perubahan revolusi

industri 4.0 dengan memperbarui website resminya guna memberikan informasi terkini tentang aktivitas kampus selama pandemi. Penelitian ini bertujuan untuk menguji dan menganalisis tingkat keamanan website resmi ITP, sekaligus memberikan saran untuk mengatasi masalah keamanan berdasarkan hasil analisis. Pengujian keamanan website dilakukan menggunakan Acunetix Vulnerability Scanner. Data yang diperoleh dianalisis menggunakan metode deskriptif, yang menyajikan hasil analisis dalam bentuk narasi untuk memberikan kejelasan tentang kondisi keamanan website. Hasil pengujian awal menunjukkan bahwa website ITP berada pada threat level 3, yang tergolong kategori high. Sebanyak 714 celah keamanan ditemukan, terdiri dari 94 celah pada level high, 25 pada level medium, 46 pada level low, dan 549 pada level informational.

Penelitian yang dilakukan oleh (Albahar et al., 2022) berfokus pada bagaimana cara mendeteksi kerentanan kompleks tanpa menghasilkan banyak false positives, yang membutuhkan validasi manual. Penelitian ini menbandingkan alat pen-testing untuk deteksi kerentanan aplikasi web dengan kriteria benchmarking, mencakup alat seperti Burp Suite, OWASP ZAP, dan Fortify WebInspect. Burp Suite terbukti unggul dalam cakupan deteksi OWASP Top 10 untuk alat komersial, sedangkan OWASP ZAP menunjukkan hasil terbaik di kategori non-komersial.

Penelitian yang dilakukan oleh (Tudela et al., 2020) mengambil topik penggabungan dari Static Analysis Security Testing, Dynamic Analysis Security Testing, dan Interactive Analysis Security Testing. Masalah dari penelitian ini yaitu banyaknya perusahaan yang menemukan developers yang seringkali kurang memiliki pengetahuan tentang keamanan, sehingga meningkatkan risiko pengembangan aplikasi yang rentan terhadap serangan. Hasil dari penelitian menunjukan bahwa penggabungan metode ini secara efektif dapat mendeteksi kerentanan keamanan sekaligus mengurangi jumlah false positive.

(Sadqi & Yassine, 2022) telah melakukan penelitian dalam bidang keamanan aplikasi web, dan menggunakan metode Dynamic Application

Security Testing (DAST) dengan pendekatan black box untuk mendeteksi celah keamanan tanpa melihat kode sumber. Penelitian ini membandingkan berbagai alat uji keamanan aplikasi web, baik open-source maupun komersial, seperti OWASP ZAP, BurpSuite, dan Acunetix. Tools BurpSuite, menunjukkan keandalan lebih tinggi dalam mendeteksi ancaman dengan tingkat *false positive* yang rendah dibandingkan dengan *tools* lainnya.

Tabel 2.1 Hasil penelitian terdahulu

No	Penulis	Judul	Hasil Penelitian
1	(Yunanri, 2023)	Analisis Keamanan Website Sistem Informasi Administrasi Kependudukan Menggunakan Metode *Vulnerability Assesment*	Hasil dari penelitian ini menunjukan bahwa website sistem informasi administrasi kependudukan memiliki beberapa peringatan keamanan dengan risiko tinggi.
2	(Mulyanto et al., 2022)	Analisis Keamanan Website SMA Negeri 2 Sumbawa Besar Menggunakan Metode *Penetration Testing* (Pentest)	Hasil dari penelitian ini menemukan beberapa celah keamanan pada website SMA Negeri 2 Sumbawa Besar yang terdeteksi 13 *sub file vulnerability* dengan status *low* dan *medium*.
3	(Wicaksono et al., 2020)	Pengujian Celah Keamanan Aplikasi Berbasis Web Menggunakan Teknik *Penetration Testing* Dan DAST (*Dynamic Application Security Testing*)	Hasil penelitian menunjukkan bahwa celah keamanan *Broken Access Control* dapat dicegah dengan membuat kode (id) yang sulit untuk ditebak, dan *Sql Injection* dapat dicegah dengan menggunakan *enscape()* pada saat pencarian basisdata.
4	(Alviansyah & Ramadhani, 2021)	Implementasi Dynamic Application Security Testing pada Aplikasi Berbasis Android	Penelitian ini menunjukan MobSF secara efektif mendeteksi berbagai celah keamanan pada aplikasi dengan metode pengujian secara dinamis.
5	(Ariyadi et al., 2023)	Analisis Kerentanan Keamanan Sistem Informasi Akademik Universitas Bina Darma Menggunakan OWASP	Hasil penelitian mengungkapkan informasi mengenai port-port yang terbuka, yang berpotensi menjadi titik masuk serangan. Selain itu, solusi untuk meningkatkan keamanan aplikasi web juga disajikan, termasuk langkah-langkah untuk meminimalkan serangan yang mungkin dilakukan oleh peretas.
6	(Zirwan, 2022)	Pengujian dan Analisis Kemanan Website Menggunakan *Acunetix Vulnerability Scanner*	Hasil pengujian awal menunjukkan bahwa website ITP berada pada *threat* level 3, yang tergolong kategori *high*. Sebanyak 714 celah keamanan ditemukan, terdiri dari 94 celah pada level *high*, 25 pada level *medium*, 46 pada level *low*, dan 549 pada level *informational*.
7	(Albahar et al., 2022)	*An Empirical Comparison of Pen-Testing Tools for Detecting Web App*	Penelitian ini menbandingkan alat *pen-testing* untuk deteksi kerentanan aplikasi web dengan kriteria *benchmarking* yang komprehensif, mencakup alat seperti Burp

		Vulnerabilities	Suite, OWASP ZAP, dan Fortify WebInspect. Burp Suite terbukti unggul dalam cakupan deteksi OWASP Top 10 untuk alat komersial, sedangkan OWASP ZAP menunjukkan hasil terbaik di kategori non-komersial.
8	(Tudela et al., 2020)	*On combining static, dynamic and interactive analysis security testing tools to improve owasp top ten security vulnerability detection in web applications*	Hasil dari penelitian menunjukan bahwa penggabungan metode ini secara efektif dapat mendeteksi kerentanan keamanan sekaligus mengurangi jumlah *false positive.*
9	(Sadqi & Yassine, 2022)	*A systematic review and taxonomy of web applications threats*	Penelitian ini membandingkan berbagai alat uji keamanan aplikasi web, baik *open-source* maupun komersial, seperti OWASP ZAP, BurpSuite, dan Acunetix. Tools BurpSuite, menunjukkan keandalan lebih tinggi dalam mendeteksi ancaman dengan tingkat *false positive* yang rendah dibandingkan dengan *tools* lainnya.

B. Landasan Teori

1. Analisis

Analisis adalah proses memecah suatu hal menjadi bagian-bagian yang lebih kecil untuk kemudian dipelajari dan dipahami hubungan antar bagian tersebut sehingga bisa mendapatkan makna yang lebih mendalam. Jadi, analisis dapat diartikan sebagai proses menguraikan atau memecah suatu masalah atau fenomena menjadi bagian-bagian yang lebih kecil untuk kemudian dipelajari dan dipahami secara mendalam.

Tujuan dari analisis adalah untuk mendapatkan pemahaman yang lebih baik tentang suatu objek atau kejadian, serta untuk menemukan pola, hubungan, atau penyebab dari suatu masalah.

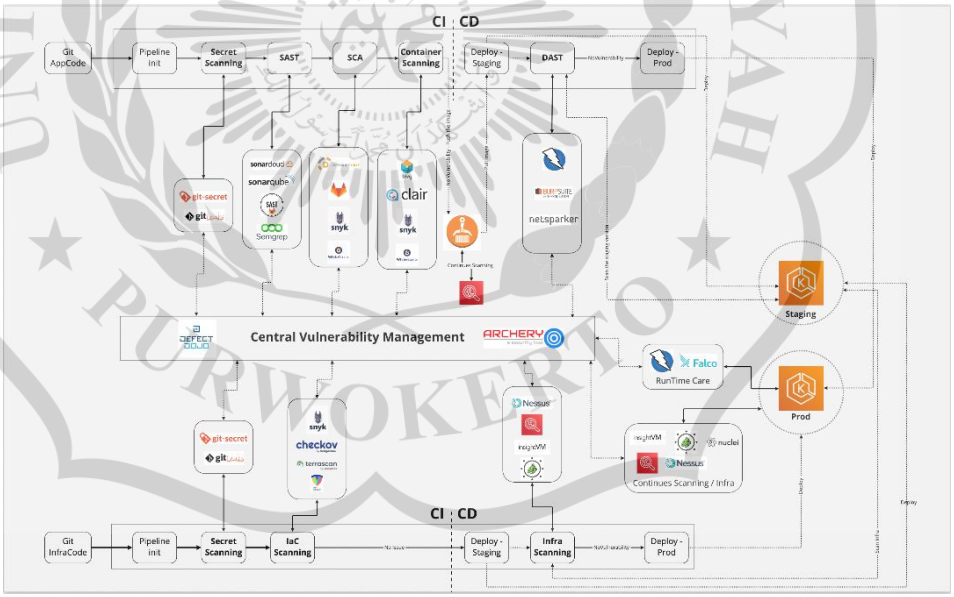
2. Dynamic Application Security Testing

Dynamic Application Security Testing (DAST) adalah metode pengujian keamanan yang memeriksa kerentanan pada aplikasi web saat sedang berjalan, dengan cara meniru serangan seperti yang dilakukan peretas. Berbeda dengan pengujian statis atau *Static Application*

Security Testing (SAST), DAST membantu tim keamanan menemukan celah yang benar-benar berisiko dan bisa dimanfaatkan oleh penyerang.

OWASP DevSecOps Guideline - v-0.2 adalah dokumen yang memberikan panduan untuk mengintegrasikan praktik keamanan (*security*) ke dalam alur kerja *DevOps*. Dokumen tersebut merekomendasikan DAST sebagai salah satu komponen kunci dalam praktik DecSecOps. Tujuannya adalah untuk mencapai *"Security as Code"* di mana keamanan tidak hanya menjadi fase akhir, tetapi tertanam di seluruh siklus pengembangan (SDLC). Dokumen ini mencakup:

a. Otomatisasi keamanan dalam CI/CD pipeline.
b. Shift-left security: Memindahkan pengujian keamanan ke fase awal pengembangan.
c. Penggunaan tools keamanan seperti SAST, DAST, dan SCA.



Gambar 2.1 Peranan DAST dalam dokumen OWASP DevSecOps

Pada gambar 2.1 di atas, gambar tersebut menunjukkan dua alur pipeline utama: satu untuk kode aplikasi (AppCode) di bagian atas, dan satu untuk kode infrastruktur (InfraCode) di bagian bawah. DAST secara spesifik ditempatkan dalam alur CI/CD untuk AppCode. Dalam

panduan DevSecOps, berbagai *tools* dan praktik keamanan direkomendasikan untuk diotomatisasi dan diintegrasikan ke dalam berbagai tahapan pipeline CI/CD (Continuous Integration/Continuous Delivery).

DAST (Dynamic Application Security Testing) adalah salah satu praktik yang sangat sering direkomendasikan dalam konteks DevSecOps. Meskipun DAST bukan "bersumber" langsung dari guideline ini, dokumen tersebut merekomendasikan DAST sebagai salah satu komponen kunci dalam praktik DevSecOps. Berikut poin-poin utamanya:

a. Integrasi DAST dalam CI/CD Pipeline

DAST muncul setelah tahap "Deploy - Staging", artinya, sebelum DAST dapat dijalankan, kode aplikasi (AppCode) harus sudah melalui serangkaian tahap sebelumnya:

1) *Pipeline Init* (Inisiasi Pipeline)
2) Secret Scanning (Pemindaian kredensial atau rahasia yang mungkin terekspos dalam kode)
3) SAST (Static Application Security Testing) (Analisis kode sumber untuk mencari kerentanan tanpa menjalankan aplikasi)
4) SCA (Software Composition Analysis) (Pemindaian untuk kerentanan dalam dependensi atau library pihak ketiga)
5) Container Scanning (Pemindaian image kontainer untuk kerentanan)
6) Push the image (Gambar kontainer yang sudah dipindai diunggah ke registry)
7) Deploy - Staging (Aplikasi di-deploy ke lingkungan Staging, yaitu lingkungan yang semirip mungkin dengan produksi, untuk pengujian akhir).

Penempatan DAST setelah *deploy* ke Staging menunjukkan bahwa DAST menguji aplikasi yang sedang berjalan secara aktif (dinamis).

b. Proses DAST

Pada tahap ini (ditandai dengan label "DAST" dan ikon *tools* seperti OWASP ZAP, Netsparker, Burp Suite), *tool* DAST akan secara otomatis mensimulasikan serangan terhadap aplikasi yang berjalan di Staging ("Scan the Staging stream"). *Tools* akan mencoba mengirimkan berbagai jenis *payload* berbahaya, memanipulasi parameter, dan menganalisis respons dari aplikasi untuk mengidentifikasi kerentanan runtime seperti SQL Injection, Cross-Site Scripting (XSS), masalah konfigurasi server, celah logika bisnis yang terekspos, dll. DAST beroperasi dari perspektif "black-box", artinya ia menguji aplikasi dari luar tanpa pengetahuan tentang kode sumber internalnya, mirip seperti bagaimana penyerang akan melihat aplikasi.

c. Output dari Tahap DAST dan Integrasi Selanjutnya:

Hasil utama dari pemindaian DAST adalah laporan kerentanan yang ditemukan. Dalam diagram, panah dari DAST mengarah ke "Central Vulnerability Management" (Manajemen Kerentanan Terpusat), dengan contoh tool Archery. Ini berarti semua temuan kerentanan dari DAST (dan juga dari SAST, SCA, dll.) dikumpulkan dan dikelola di satu tempat. Hal ini memudahkan pelacakan, prioritisasi, dan penugasan perbaikan kerentanan.

Setelah tahap DAST, ada jalur yang mengarah ke "Deploy - Prod" (Deployment ke Produksi). Ini menyiratkan bahwa hasil DAST menjadi salah satu penentu (quality gate) apakah aplikasi dianggap cukup aman untuk dilanjutkan ke lingkungan produksi. Jika DAST menemukan kerentanan kritis, pipeline dapat dihentikan, atau alert akan dikirim ke tim pengembang untuk

segera diperbaiki. Jika DAST tidak menemukan kerentanan kritis atau semua kerentanan yang ditemukan berada dalam batas toleransi risiko yang diterima (atau sudah ditangani), maka aplikasi dapat melanjutkan ke tahap deployment produksi.

Berdasarkan panduan dokumen OWASP DevSecOps Guideline - v-0.2, pada penelitian ini akan mengimplementasikan DAST menggunakan tools OWASP ZAP, dan Codename SCNR dalam melakukan pengujiannya di lingkungan *production,* atau aplikasi yang sudah berjalan melalui serangkai proses dalam SDLC.

3. OWASP ZAP

OWASP ZAP (Zed Attack Proxy) adalah sebuah alat *open-source* yang sangat populer digunakan untuk menguji keamanan aplikasi web. Alat ini dirancang untuk membantu para pengembang dan penguji untuk menemukan kerentanan keamanan pada aplikasi web mereka sebelum dieksploitasi oleh *hacker*. ZAP bekerja dengan cara mensimulasikan serangan yang umum dilakukan oleh *hacker*, seperti injeksi SQL, *cross-site scripting* (XSS), dan lainnya.

OWASP ZAP merupakah salah satu tools yang di rekomendasikan oleh OWASP DevSecOps Guideline - v-0.2, sebagai salah satu tools yang *open-source* yang ditawarkan oleh OWASP untuk melakukan pengujian keamanan.

4. Codename SCNR

SCNR sering disebut sebagai penerus atau pengganti dari Arachni, merupakan sebuah pemindai keamanan web open-source yang populer sebelumnya. SCNR bertujuan untuk menjadi lebih efisien, lebih cepat (disebutkan sekitar 6+ kali lebih cepat dari Arachni), dan lebih hemat sumber daya (bandwidth dan RAM) dibandingkan pendahulunya.

Codename SCNR, yang dikembangkan oleh Ecsypno, diposisikan sebagai sebuah *framework* pemindai keamanan aplikasi web DAST yang modern, modular, terdistribusi, dan berkinerja tinggi. Dirancang untuk mampu menganalisis perilaku dan keamanan aplikasi

web modern yang kompleks, termasuk yang banyak menggunakan JavaScript, DOM manipulation, HTML5, dan AJAX. Ia didukung oleh teknologi seperti Google Chrome untuk memastikan kompatibilitas dengan teknologi web terkini.

SCNR menawarkan berbagai edisi, mulai dari *Free 30-day Trial*, *Community* (dengan fungsionalitas terbatas seperti CLI tanpa *authenticated scans*, *reporting*, atau API), hingga edisi berbayar (Basic, Pro, SDLC, Enterprise) yang menawarkan fitur lebih lengkap seperti pemindaian terautentikasi, *reporting*, Web UI, REST API, agen jarak jauh, dan penjadwalan.

5. Back-box Testing

   Black-box testing adalah pendekatan pengujian perangkat lunak yang dilakukan tanpa mengetahui struktur internal, desain, atau implementasi kode dari perangkat lunak tersebut. Pengujian ini berfokus pada fungsionalitas perangkat lunak dari sudut pandang pengguna akhir atau pengguna yang memakainya.

6. White-box Testing

   White-box testing adalah metode pengujian di mana penguji memiliki pengetahuan tentang struktur internal, desain, dan implementasi kode dari sistem.

7. Cross-Site Scripting (XSS)

   Serangan ini melibatkan penyisipan script berbahaya ke dalam halaman web yang dapat dieksekusi di browser pengguna dan memungkinkan manipulasi tampilan website.

8. SQL Injection

   Serangan ini terjadi ketika penyerang menyisipkan perintah *sql* berbahaya melalui *input* aplikasi dan memungkinkan penyerang untuk mengakses, mengubah, atau menghapus data di database.

9. True Positive

   Hasil pengujian dengan benar mendetetksi adanya masalah kerentanan, dan setelah di evaluasi lebih lanjut memang benar terdapat

adanya kerentanan tersebut. Sebagai contoh tool keamanan mendeteksi adanya *SQL injection* pada halaman login, dan setelah di periksa, kerentanan tersebut benar ada.

10. False Positive

    Hasil pengujian keliru mendeteksi adanya kerentanan, padahal sebenarnya itu tidak ada. Sebagai contoh *tool* keamanan mendeteksi adanya *cross-site scripting* (XSS) di suatu input, tetapi setelah di cek input tersebut sudah di sanitasi dengan benar dan tidak menunjukan adanya kerentanan.

11. True Negative

    Hasil pengujian benar dan mengkonfirmasi tidak ada masalah, dan setelah di cek lebih lanjut memang tidak di temukan masalah. Sebagai contoh *tools* pemindaian tidak melaporkan kerentanan di halaman statis, dan setelah diverifikasi, halaman tersebut memang aman.

12. False Negative

    Hasil pengujian gagal dalam mendeteksi suatu masalah, padahal masalah tersebut memang benar adanya. Pemindai tidak melaporkan adanya kerentanan, padahal sebenarnya ada kerentanan nyata yang terlewat dan tidak terdeteksi.