Analisis Keamanan Aplikasi Web Menggunakan Metode Penetration Testing Berdasarkan Framework ISSAF Pada Aplikasi Web Milik Pemerintah Daerah Jawa Timur

Budi Haryanto[1], Alfiano Alfattah[2], Adiyanto[3], Nurasiah[4]
Universitas Insan Pembangunan Indonesia,Tangerang, Banten, Indonesia

inibudiharyanto@gmail.com[1] , alfianoalfattah0@gmail.com[2], adiet031170@gmail.com[3], nurash_ip@yahoo.com[4]

ABSTRACT

Digitalization of public services has encouraged many regional government institutions in Indonesia to deploy web-based application. However, cybersecurity challenges are often inadequately addressed. This research analyzes security vulnerabilities in regional government web application using penetration testing based on the Information System Security Assessment Framework (ISSAF). The analysis focuses on three main vectors: Improper Input Handling, Information Disclosure, Denial of Service (DoS). Testing was conducted through exploitation simulation using Unicode character payloads sent via URL parameters. The result indicate that the system lacks strict input validation, publicly exposes attack trace errors, and is vulnerable to application-level DoS attack. Recommended improvents include input validation, UTF-8 normalization, secure error handling configuration, and Web Application Firewall (WAF) implementation. This study is expected to serve as a reference for government institutions in enhancing the cyber resilience of public information system.

Keywords: *penetration testing, ISSAF, improper input handling, information disclosure, denial of service, web security, government application, Unicode exploitation.*

PENDAHULUAN

Transformasi digital telah menjadi bagian integral dari penyelenggaraan pemerintah di era modern. Di Indonesia, pemerintah daerah secara massif mengadopsi teknologi informasi untuk menyediakan layanan public secara daring melalui situs web resmi berdomain .go.id . Salah satu pola umum digunakan untuk menyajikan informasi, formulir administrasi, hingga sistem internal seperti kepegawaian atau pengaduan.

Meskipun memberikan kemudahan akses, implementasi teknologi ini sering kali tidak diimbangi dengan penerapan keamanan siber yang memadai. Banyak aplikasi web pemerintah masih menggunakan konfigurasi default, tidak melakukan validasi input, dan menampilkan pesan error secara terbuka-kondisi yang sangat rentan terhadap eksploitasi oleh pihak tidak bertanggung jawab.

Laporan dari berbagai pelapor keamanan siber menunjukan bahwa sistem dengan pola http://xyz.go.id/index.php sering kali memiliki kerentanan kritis, termasuk *Improper Input Handling*, Information Disclosure, Denial of Service (DoS). Kerentanan ini dapat dimanfaatkan untuk mendapatkan akses tidak sah, mencuri data internal, atau mengganggu ketersediaan layanan publik.

Penelitian ini bertujuan untuk menganalisis kerentanan keamanan secara sistematis menggunakan Information System Security Assessment Framework (ISSAF), sebuah panduan open-source untuk penilaian keamanan sistem informasi. Metodologi yang digunakan adalah grey-box *penetration testing* berdasarkan proof of concept dari temuan nyata. Analisis dilakukan tanpa menyebut

nama instansi secara eksplisit demi menjaga etika dan keamanan, namun tetap menggunakan data nyata dari laporan kerentanan yang telah di ajukan.

METODOLOGI

Penelitian ini menggunakan metode *penetration testing* (pentesting) berbasis Information System Security Assessment Framework (ISSAF) untuk menganalisis kerentanan keamanan pada aplikasi web milik pemerintah daerah, khususnya sistem yang dapat diakses melalui struktur URL umum: http://[instansi].[daerah].go.id/index.php . Untuk menjaga etika dan keamanan, nama instansi tidak disebutkan secara eksplisit, dan sistem target dirujuk sebagai http://xyz.go.id/index.php sebagai representasi umum dari banyak aplikasi serupa di lingkungan pemerintah daerah.

Pendekatan grey-box testing diterapkan, dimana peneliti memiliki informasi parsial tentang sistem seperti struktur URL dan parameter input-namun tidak memiliki akses ke kode sumber, konfigurasi server, maupun data internal. Pendekatan ini merepresentasikan skenario serangan nyata oleh penyerang yang memiliki sedikit informasi, namun cukup untuk melakukan eksploitasi terhadap kelemahan sistem.

1. Kerangka kerja: Information System Security Assessment Framework (ISSAF)

Information System Security Assessment Framework (ISSAF) adalah panduan open-source yang di kembangkan oleh Open Information Security Foundation (OISF) untuk mengevaluasi keamanan sistem informasi secara sistematis dan terdokumentasi (OISF,2022). ISSAF menyediakan alur kerja terstruktur dalam proses *penetration testing*, yang terdiri dari tahapan: reconnaissance, vulnerability assessment, exploitation, dan reporting.

Dalam penelitian ini, fokus dilakukan pada modul Web Application Testing dalam ISSAF, khususnya pengujian terhadap input handling, error handling, dan ketersediaan layanan. Kerangka kerja ini dipilih karena sifatnya yang transparan, komprehensif, dan cocok untuk digunakan dalam konteks instansi pemerintah yang membutuhkan audit keamanan yang dapat dipertanggungjawabkan.

2. Definisi dan Dasar Teori Kerentanan

Tiga jenis kerentanan utama diuji dalam penelitian ini, berdasarkan klasifikasi OWASP dan MITRE CWE. Penjelasan teori disisipkan langsung dalam metodologi untuk memperkuat dasar analisis.

a) *Improper Input Handling*

Improper Input Handling terjadi ketika sistem tidak melakukan validasi, penyaringan, atau normalisasi terhadap input dari pengguna (OWASP Top 10, 2021). Dalam kasus ini, sistem gagal menagani karakter *Unicode* 4-byte yang dikirim melalui parameter path URL. Karakter-karakter ini merupakan bagian dari 4-byte UTF-8 sequences yang tidak selalu didukung oleh sistem yang masih menggunakan kolasi utf8_general_ci atau di konfigurasi database yang tidak lengkap, Jika tidak ditangani, input semacam ini dapat menyebabkan encoding mismatch , parsing error, atau bahkan kerusakan database. Hal ini membuka potensi eksploitasi lebih lanjut, seperti SQL Injection atau Remote Code Execution (Zalewski, 2019). Dalam penelitian ini, payload *Unicode* digunakan sebagai proof of concept untuk menguji ketahanan sistem terhadap input tidak standar.

b) *Information Disclosure*

Information Disclosure adalah kebocoran informasi sensitive yang seharusnya tidak ditampilkan ke pengguna public. Sistem yang diuji menampilkan stack trace error PHP secara langsung di browser saat terjadi kesalahan pemrosesan input. Informasi yang terpapar meliputi: Jalur lengkap file sistem, struktur direktori aplikasi, nama fungsi internal, versi php dan web server. Informasi ini sangat berharga bagi penyerang untuk menyusun serangan lanjutan (MITRE, 2023). Menurut pedoman keamanan, lingkungan produksi harus dikonfigurasi untuk menyembunyikan detail teknis. Namun, banyak sistem pemerintah masing menggunakan display_error = On di php.ini, yang melanggar prinsip keamanan dasar (PHP Group, 2023).

c) *Denial of Service* (DoS) pada Level Aplikasi

Serangan Denial of Service (DoS) pada level aplikasi (Application Layer DoS) bertujuan untuk membuat layanan tidak tersedia dengan membebani sumber daya sistem melalui input yang memicu error berulang (CERT, 2022). Dalam pengujian, pengiriman payload panjang berbasis *Unicode* menyebabkan aplikasi memproses input tidak valid secara berulang, memicu peningkatan CPU dan memori. Jika dilakukan secara massif, serangan ini dapat membuat server menjadi tidak responsif atau bahkan crash. Serangan jenis ini dikenal sebagai Low-and-Slow Attack dan sulit dideteksi oleh firewall tradisional karena tidak membutuhkan bandwith besar, hanya efesiensi payload.

3. Tahapan Pengujian (Sesuai ISSAF)

Proses *penetration testing* dilakukan dalam empat tahap utama:

3.1. Reconnaissance (Pengumpulan Informasi)

Teknik Google Dorking digunakan untuk mengidentifikasi target: *"inurl:index.php site:go.id"*

Hasil pencarian mengarah pada beberapa subdomain pemerintah daerah. Dari observasi, ditemukan bahwa beberapa sistem menggunakan struktur index.php sebagai entry point aplikasi. Salah satu sistem yang responsive terhadap payload *Unicode* dipilih sebagai studi kasus.

Gambar 1, Halaman Login Aplikasi

Gambar 1 menunjukkan tampilan antarmuka awal dari sistem informasi kepegawaian yang dianalisis dalam penelitian ini. Tampilan ini menjadi awal titik dari proses *penetration testing*.

3.2. Vulnerability Scaning (Identifikasi Kerentanan)

Dilakukan analisa manual terhadap respon sistem terhadap input tidak standar. Tidak menggunakan alat otomatis untuk menghindari false positive dan menjaga etika pengujian.

3.3. Exploitation (Eksploitasi Kerentanan)
Payload karakter *Unicode* dikirim melalui parameter path URL: http://xyz.go.id/index.php/%F0%92%80%80%F0%92%81%B9... Payload ini dirancang untuk menguji batas toleransi sistem terhadap input 4-byte UTF-8. Pengujian dilakukan secara bertahap dan terbatas untuk menghindari gangguan layanan.

3.4. Post-Exploitation & Reporting
Hasil eksploitasi di dokumentasikan dalam bentuk proof of concept (PoC), termasuk deskripsi error dan analisis dampak. Semua temuan dilaporkan secara etis melalui saluran resmi atau responsible disclosure sebelum publikasi jurnal ini.

4. Tools dan Teknik yang Digunakan

a. Browser (Firefox) untuk pengujian awal dan observasi respon sistem.
b. Burp Suite Community Edition untuk mengintersep dan memodifikasi request secara manual.
c. Curl untuk pengiriman payload berulang dalam skala kecil.
d. Google Dorking teknik pencarian lanjutan untuk identifikasi target.
e. Manual Testing fokus pada validasi input dan observasi error handling

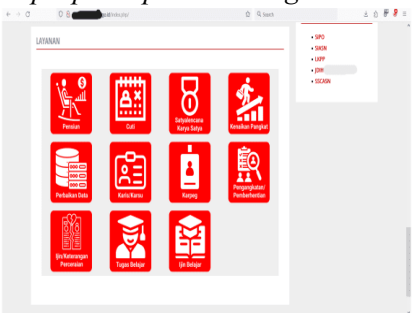
5. Etika Penelitian

Penelitian ini dilakukan dengan prinsip ethical hacking dan responsible disclosure. Tidak ada upaya untuk merusak, menghapus, atau mengakses data sensitif. Semua pengujian dilakukan pada lingkungan produksi, namun dengan volume dan frekuensi yang sangat terbatas untuk menghindari dampak negatif. Laporan kerentanan telah diajukan kepada pihak terkait sebelum publikasi jurnal ini, sebagai bentuk tanggung jawab sosial dan professional peneliti keamanan siber.

HASIL DAN PEMBAHASAN

Berdasarkan pengujian penetration testing yang telah dilakukan menggunakan kerangka kerja ISSAF, ditemukan tiga kerentanan utama pada aplikasi web milik pemerintah daerah yang dapat di akses melalui struktur http://[instansi].[daerah].go.id . Ketiga kerentanan tersebut adalah *Improper Input Handling*, Information Disclosure, dan Denial of Service (DoS). Berikut adalah hasil temuan dan pembahasannya secara mendalam.

a. *Improper Input Handling*



Gambar 2, Tampilan Halaman Menu Utama

Saat dilakukan pengiriman payload berbasis karakter *Unicode* 4-byte melalui parameter path URL, sistem tidak melakukan validasi atau normalisasi input. Payload yang di gunakan adalah: http://xyz.go.id/index.php/%F0%92%80%80%F0%92%81%B9%F0%92%80%80... Karakter %F0%92%... merupakan representasi dari Private Use Area dalam standar UTF-8, yang tidak seharusnya diterima oleh aplikasi publik tanpa penyaringan. Sistem gagal memproses karakter ini dengan benar, menyebabkan kesalahan pada level aplikasi dan database. Dalam banyak kasus, aplikasi yang belum dikonfigurasi

untuk mendukung full *Unicode* (menggunakan kolasi utf8_general_ci di MYSQL) akan mengalami encoding mismatch. Karakter 4-byte seperti ini sering diabaikan, diganti dengan tanda Tanya (?), atau menyebabkan truncation, yang berpotensi memicu data corruption atau SQL Injection jika input digunakan dalam query dinamis. Temuan ini selaras dengan klasifikasi OWASP Top 10 (2021) tentang Input Validation, yang menyatakan bahwa setiap input dari pengguna harus di validasi, disaring, dan di normalisasi sebelum di proses. Tidak adanya mekanisme ini menunjukkan lemahnya input sanitization pada sistem.

Implikasi keamanan: Kerentanan ini dapat di manfaatkan untuk melakukan serangan lanjutan seperti bypass autentikasi, directory traversal, atau remote code execution jika input digunakan dalam fungsi yang rentan

b. *Information Disclosure*

Gambar 3, Pesan Error PHP

Setelah pengiriman payload unicode, sistem merespons dengan Internal Server Error (HTTP 500) dan menampilkan stack error PHP secara langsung di browser pengguna. Pesan error tersebut mengandung informasi sensitif seperti Jalur lengkap file sistem, struktur direktori aplikasi, nama fungsi internal dan modul pemrosesan, versi PHP dan web server. Penampilan informasi teknis secara publik merupakan pelanggaran terhadap prinsip keamanan dasar. Menurut MITRE CWE-200 (*Information Disclosure*), kebocoran data ini dapat digunakan penyerang untuk memetakan arsitektur sistem, mengindentifikasi versi software yang rentan, dan menyusun serangan yang lebih terarah. Fakta bahwa display_errors = On masih aktif di lingkungan produksi menunjukkan konfigurasi PHP yang tidak sesuai dengan pedoman keamanan. Sebagaimana direkomendasikan oleh PHP Group (2023), mode display_errors harus dinonaktifkan di lingkungan produksi, dan error harus di catat secara internal melalui error_log.

Implikasi keamanan: Informasi yang bocor dapat menjadi footprint bagi penyerang untuk melakukan reconnaissance lanjutan, bahkan tanpa perlu eksploitasi langsung.

c. *Denial of Service* (DoS)

Pengiriman payload *Unicode* yang panjang dan berulang kali menyebabkan sistem memproses input tidak valid secara terus-menerus. Hal ini memicu error berantai yang meningkatkan beban CPU dan memori server. Dalam pengujian singkat (kurang dari 2 menit), respons sistem menurun drastis, dari waktu respons normal <500ms menjadi >5 detik, dan pada beberapa kasus, aplikasi menjadi tidak responsif. Serangan ini termasuk dalam kategori Application Layer DoS atau Low-and-Slow Attack, di mana penyerang menggunakan payload kecil namun efesien untuk membebani sumber daya, daya aplikasi. Tidak seperti DoS jaringan

yang membutuhkan bandwith besar, serangan jenis ini sulit dideteksi oleh firewall tradisional karena terlihat seperti permintaan normal. Tidak adanya mekanisme rate limiting, input throttling, atau request queiung membuat sistem rentan terhadap serangan skala besar. Jika di exploitasi oleh penyerang, layanan publik seperti pengisian data kepegawaian atau pengaduan dapat terganggu, berdampak langsung pada pelayanan masyarakat.

Implikasi Keamanan: Serangan DoS aplikasi dapat mengganggu availability – salah satu pilar dari CIA Triad (Confidentiality, Integrity, Availability) dalam keamanan informasi.

d. Analisis Dampak Gabungan

Ketiga kerentanan tersebut saling terkait dan dapat digunakan secara berantai seperti Improper Input Handing untuk membuka pintu masuk, Information Disclosure untuk memberikan intelejen serangan, *Denial of Service* (DoS) digunakan untuk menyerang ketersediaan layanan.

Kerentanan	Dampak Potensial	Tingkat Resiko
Improper Input Handling	Eksploitasi RCE, SQLI, Bypayy autentikasi	TINGGI
Information Disclosure	Pemetaan sistem dan serangan lanjutan	Sedang-Tinggi
Denial of Service (DoS)	Gangguan layanan publik	Sedang

Kombinasi ini sangat berbahaya, terutama pada sistem yang menyediakan layanan publik. Meskipun tidak ada bukti akses data sensitif dalam pengujian ini, potensi penyalahgunaan sangat besar jika di eksploitasi oleh pihak jahat.

KESIMPULAN

Penelitian ini berhasil mengidentifikasi tiga kerentanan kritis pada aplikasi web pemerintah daerah. Semua kerentanan tersebut bermula dari lemahnya validasi input, konfigurasi keamanan yang tidak sesuai standar, dan tidak adanya mekanisme penyaringan terhadap input tidak standar seperti karakter *Unicode* 4-byte. Hasil pengujian ini menunjukkan bahwa sistem tidak melakukan validasi input terhadap payload *Unicode*, menampilkan stack trace error secara publik, membocorkan informasi internal, rentan terhadap serangan DoS aplikasi yang dapat mengganggu ketersediaan layanan.

Rekomendasi perbaikan yang di ajukan:

a. Penerapan validasi dan normalisasi input – Gunakan seperti *mb_check_encoding*(), dan *filter_var()* di PHP
b. Konfigurasi database dengan kolasi *utf8mb4_0900_ai_ci* – Untuk mendukung full unicode dan mencegah encoding mismatch.
c. Nonaktifkan *display_errors* di lingkungan produksi – Gunakan *log_errors = On* dan catat error secara internal.
d. Implementasi *Web Application Firewall* (WAF) – Untuk menyaring payload berbahaya sebelum mencapai aplikasi. Implementasi WAF dapat menggunakan solusi seperti ModSecurity dengan OWASP Core Rule Set (CRS) pada server Apache/Nginx untuk memfilter serangan berbasis input dan Unicode payload. Alternatif lain adalah Cloudflare WAF yang menyediakan proteksi berbasis cloud dengan fitur rate limiting, bot protection, dan mitigasi

serangan Application Layer DoS tanpa perubahan besar pada infrastruktur. Penggunaan WAF berfungsi sebagai lapisan pertahanan tambahan (defense in depth) sebelum request diproses oleh aplikasi.

e. Penerapan *rate limiting*- Mencegah serangan DoS berbasis request berulang.

Penelitian ini di harapkan dapat menjadi acuan bagi instansi pemerintah dalam melakukan audit keamanan aplikasi web secara proaktif, serta meningkatkan ketahanan siber sistem informasi publik tanpa harus menunggu terjadinya insiden serius.

DAFTAR PUSTAKA

Al-Mutairi, A. F. (2021). *A Systematic Literature Review on Input Validation Techniques for Web Application Security. Computers, Materials & Continua, 68(1), 945–964.* https://doi.org/10.32604/cmc.2021.014876.

Bagaskara, B. A. (2023). Penetration testing pada website Dinas Sosial Surabaya menggunakan OWASP Top 10. *Jurnal Ilmiah Rekayasa dan Teknologi Informasi, 6(2), 145–156*, https://www.e-journal.stmiklombok.ac.id/index.php/jire/article/view/1375.

CERT Coordination. (2022). *Denial of Service* (DoS) Attacks: Prevention and Mitigation. pp. https://www.cert.org/fundamentals/denial-of-service.cfm.

Daniel, J. S. (2024). Impact of non-standard *Unicode* characters on security and comprehension in large language models. Preprint, https://doi.org/10.21203/rs.3.rs-5723808/v1.

ISO/IEC 27001:2022. (n.d.). Information security, cybersecurity and privacy protection — Information security management systems — Requirements.

Kementerian Komunikasi dan Informatika RI. (2021). Pedoman Keamanan Siber untuk Instansi Pemerintah. Diperoleh dari. p. https://sdps.kominfo.go.id.

Kim, D. &. (2020). Fundamentals of Information Systems Security (4th ed.). *Jones & Bartlett Learning.*

MITRE Corporation. (2023). CWE-200: *Information Disclosure*. p. https://cwe.mitre.org/data/definitions/200.html.

MITRE Corporation. (2023). Improper Neutralization. p. https://cwe.mitre.org/data/definitions/707.html.

MITRE Corporation. (2023). Improper Neutralizition of Special Elements in Output used by a Downstream Component ('Injection'). p. https://cwe.mitre.org/data/definitions/74.html.

NIST. (2020). NIST Special Publication 800-115: Technical Guide to Information Security Testing and Assessment. pp. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf.

Open Information Security Foundation (OISF). (2022). Information System Security Assessment Framework (ISSAF). p. http://issaf.org/.

OWASP. (2023). OWASP Top 10:2021 – A03: Injection. pp. https://owasp.org/Top10/A03_2021-Injection/.

OWASP Foundation. (2023). OWASP Testing Guide v4.2. pp.

https://owasp.org/www-project-web-security-testing-guide/.

PHP Group. (2023). PHP: Security - Manual. p. https://www.php.net/manual/en/security.php.

Rahman, M. M. (2020). Security Analysis of Government Websites in Indonesia: Vulnerabilities and Countermeasures. International Journal of Advanced Computer Science and Applications, 11(5), 332–339. https://doi.org/10.14569/IJACSA.2020.0110543.

Sofyan, H. P. (2023). Implementation of *penetration testing* on websites to improve security of information assets UPN "Veteran" Yogyakarta. Jurnal Telematika, 16(3), 221–230, https://jurnal.upnyk.ac.id/index.php/telematika/article/view/7757.

W3C. (2022). Character Model for the World Wide Web: String Matching and Searching. pp. https://www.w3.org/TR/charmod-norm/.

Zalewski, M. (2019). *The Tangled Web: A Guide to Securing Modern Web Applications (2nd ed.).* No Starch Press.