Department of Digital Business
Journal of Artificial Intelligence and Digital Business (RIGGS)
Homepage: https://journal.ilmudata.co.id/index.php/RIGGS
Vol. 5 No. 2 (2026) pp: 19671- 19677
P-ISSN: 2963-9298, e-ISSN: 2963-914X

Penerapan Web Application Firewall Berbasis Mod Security untuk Mengatasi Server Side Request Forgery (SSRF)

Lalu Amirulloh Taufikurrahman[1], I Putu Hariyadi[2], Ondi Asroni[3]
[1,2,3]Program Studi Teknologi Informasi, Fakultas Teknik, Universitas Bumigora
[1]laluamirullohtaufikurrahman@gmail.com, [2]putu.hariyadi@universitasbumigora.ac.id,
[3]ondisembahulun@universitasbumigora.ac.id

Abstrak

Keamanan aplikasi web menjadi aspek krusial di tengah meningkatnya ancaman siber, khususnya terhadap kerentanan logika bisnis seperti Server-Side Request Forgery (SSRF) dan Open Redirect. Penelitian ini bertujuan untuk menganalisis efektivitas Web Application Firewall (WAF) Mod Security dengan aturan OWASP Core Rule Set (CRS) dalam memitigasi kerentanan tersebut pada sebuah aplikasi web target. Metodologi yang digunakan dalam penelitian ini adalah metode eksperimen yang diintegrasikan dengan kerangka kerja Network Development Life Cycle (NDLC). Metodologi NDLC adalah kerangka kerja yang digunakan untuk merancang, membangun, dan mengelola infrastruktur jaringan atau keamanan secara terstruktur. Hasil pengujian tahap awal menunjukkan bahwa mode blocking dengan aturan standar efektif menggagalkan serangan dengan merespon status HTTP 403 Forbidden. Namun, konfigurasi standar ini memicu insiden False Positive, di mana permintaan valid yang memuat alamat IP internal pada fitur Check Stock turut terblokir, sehingga mengganggu ketersediaan layanan (availability). Untuk mengatasi hal tersebut, dilakukan penyesuaian aturan (Rule Tuning) melalui mekanisme whitelisting berbasis Regular Expression (Regex) yang divalidasi secara ketat. Hasil verifikasi akhir menunjukkan bahwa implementasi aturan custom berhasil memulihkan akses untuk permintaan yang sah (HTTP 200 OK) sekaligus tetap memblokir upaya eksploitasi dan manipulasi parameter. Penelitian ini menyimpulkan bahwa implementasi Mod Security dengan kombinasi OWASP CRS dan Custom Rules mampu melindungi aplikasi pada server dari serangan SSRF secara efektif, menjaga fungsionalitas katalog tetap berjalan normal, serta memungkinkan administrator memantau lalu lintas data yang mencurigakan.

Kata Kunci: *Web Application Firewall, Mod Security, OWASP CRS, SSRF, Open Redirect, Insecure Design, Owasp Top 10. .*

1. Latar Belakang

Perkembangan teknologi aplikasi *web* berbasis *PHP* yang dinamis membawa tantangan besar pada aspek keamanan siber, terutama terkait munculnya celah keamanan akibat kelalaian dalam desain (*insecure design*)[1][2]. Salah satu jenis ancaman yang paling krusial dan merusak pada infrastruktur *web* modern saat ini adalah serangan *Server-Side Request Forgery* (SSRF)[1][3][4]. Melalui eksploitasi kerentanan *SSRF*, penyerang dapat memanipulasi *server* aplikasi untuk melakukan permintaan HTTP tidak sah ke dalam jaringan internal yang terisolasi maupun ke jaringan luar[1][5]. Dampak dari serangan ini sangat destruktif bagi kelangsungan sistem, karena dapat mengakibatkan terjadinya kebocoran data sensitif (*information disclosure*) seperti terekspos nya kredensial yang digunakan untuk login, hingga memicu eksekusi kode jarak jauh (*remote code execution*) yang mampu mengambil alih kendali *server* secara penuh[1][6].

Penelitian terdahulu telah berupaya mengatasi ancaman ini melalui berbagai pendekatan, baik berbasis alat deteksi otomatis, pengujian penetrasi, maupun kecerdasan buatan[7][8][9][10][11]. Pada aspek pengembangan alat deteksi otomatis untuk mengatasi tipe data dinamis *PHP*, peneliti menciptakan alat *taint analysis* bernama *Artemis* yang berhasil menemukan 207 jalur rentan (106 *SSRF* sejati) dengan 15 *false positive* dan 35 *SSRF* baru dari uji coba terhadap 250 aplikasi *PHP*[7]. Selain itu, dikembangkan pula prototipe *SSRFuzz* yang menggunakan metode *fuzzing* pada 27 aplikasi dunia nyata termasuk *WordPress* dan *Joomla*, yang sukses mengidentifikasi 28 celah *SSRF* (25 belum dilaporkan sebelumnya) dan menghasilkan 16 *ID CVE* baru[1]. Dari sudut pandang dampak serangan melalui pengujian menggunakan *Penetration Testing Execution Standard* (*PTES*) pada *buggy web application*, ditemukan bahwa *insecure design* pada aplikasi dapat memicu *information disclosure* berupa kebocoran kredensial *login* hingga *remote code execution* (RCE) untuk mengambil data *server*[12]. Sementara itu,

untuk deteksi adaptif berbasis kecerdasan buatan, penggunaan teknik *Deep Learning* dengan arsitektur *Long Short-Term Memory* (*LSTM*) terbukti sangat tangguh karena mampu menghasilkan akurasi deteksi serangan sebesar 96% pada pengujian jenis serangan SSRF, serta mencapai akurasi hingga 98,2% pada analisis empiris lainnya yang terbukti mengungguli model bahasa *BERT* yang hanya mencapai akurasi 94%[13].

Meskipun metode deteksi otomatis dan *Deep Learning* di atas sangat efektif dalam mengidentifikasi celah keamanan, masih terdapat masalah dan celah penelitian (*research gap*) yang signifikan pada aspek mitigasi *preventif* di lingkungan produksi menggunakan infrastruktur standar industri[14]. Penelitian dengan metode *PTES* sebelumnya memang menyarankan penambahan "konfigurasi tertentu", namun belum menjabarkan secara teknis bagaimana menangani konflik antara keamanan sistem dan ketersediaan layanan (*security vs availability*)[12]. Di lingkungan nyata, implementasi *WAF* berbasis aturan (*rule-based*) seperti ModSecurity dengan *OWASP CRS* sering kali menghadapi kendala tingginya angka *False Positive* ketika dihadapkan pada logika bisnis aplikasi yang kompleks—sebuah tantangan operasional yang belum dijawab secara mendalam oleh penelitian berbasis AI maupun deteksi kode statis[1]. Oleh karena itu, diharapkan hadir sebuah solusi praktis yang mampu melakukan mitigasi preventif secara efektif namun tetap menjaga fungsionalitas sistem.

Sebagai jawaban atas permasalahan tersebut, penelitian ini dilakukan dengan judul **"Penerapan *Web Application Firewall* Berbasis Mod Security Untuk Mengatasi *ServerSide Request Forgery* (*SSRF*)"**. Solusi ini diuji melalui serangkaian eksperimen simulasi serangan *SSRF* langsung pada *endpoint* aplikasi katalog di lingkungan produksi dengan kondisi *WAF* yang aktif. Adapun parameter uji coba yang digunakan untuk mengukur keberhasilan penelitian ini meliputi nilai akumulasi skor anomali (*inbound anomaly score*), tingkat keberhasilan penurunan angka *False Positive* setelah penerapan penyesuaian aturan (*Rule Tuning*), serta stabilitas fungsionalitas aplikasi katalog saat memproses permintaan data.

Hasil akhir dari penelitian ini diharapkan dapat memberikan manfaat umum yang luas bagi pengembangan sistem keamanan aplikasi *web*, baik di lingkungan akademis maupun industri. Secara praktis, penelitian ini dapat menjadi panduan bagi para*administrator* jaringan dalam mengonfigurasi keamanan *server* secara optimal tanpa mengorbankan kelancaran operasional bisnis aplikasi. Selain itu, implementasi kombinasi aturan ini bermanfaat dalam meminimalkan risiko eksploitasi data akibat serangan *SSRF*, meningkatkan visibilitas pemantauan log aktivitas mencurigakan secara *real-time*, serta memastikan pengguna sah dapat mengakses aplikasi secara aman tanpa kendala pemblokiran yang keliru.

2. Metode Penelitian

Metodologi yang digunakan dalam penelitian ini adalah metode eksperimen yang diintegrasikan dengan kerangka kerja *Network Development Life Cycle* (NDLC). Metodologi *NDLC* adalah kerangka kerja yang digunakan untuk merancang, membangun, dan mengelola infrastruktur jaringan atau keamanan secara terstruktur[15]. Penggunaan metode *NDLC* bertujuan untuk mengarahkan seluruh proses penelitian agar terstruktur secara teknis, mulai dari pengidentifikasian masalah hingga pengujian infrastruktur. *NDLC* terdiri dari 6 (enam) tahapan yaitu, seperti terlihat pada gambar 1.

Dari 6 tahapan yang terdapat pada metode NDLC, penulis hanya menggunakan 3 tahapan pertama yaitu Analysis, Design, dan Simulation Prototyping. Adapun penjelasan dari setiap tahapan adalah sebagai berikut:

2.1. Analysis

Pada tahap *analysis*, penulis melakukan serangkaian aktivitas investigasi untuk menggabungkan data teoritis dari studi literatur dengan data teknis dari lingkungan aplikasi. Proses ini dibagi menjadi dua sub-tahapan utama, yaitu Pengumpulan Data yang berfokus pada akuisisi referensi ilmiah dan observasi awal, serta Analisa Data yang berfokus pada pengolahan informasi tersebut sebagai landasan dasar dalam penentuan topik dan strategi konfigurasi *Web* Application Firewall (*WAF*).

2.2. Design

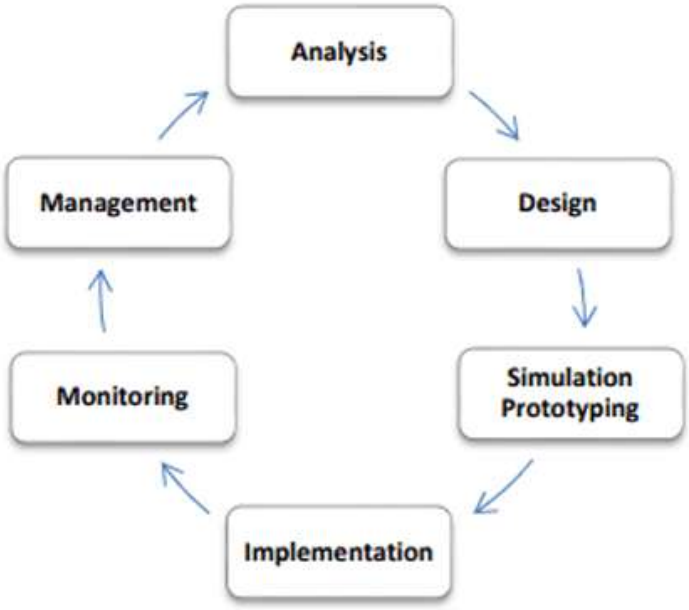
Setelah mengidentifikasi karakteristik kerentanan dan celah keamanan pada tahap sebelumnya, penulis menyusun cetak biru (*blueprint*) infrastruktur keamanan yang akan dibangun. Tujuan utama dari tahap ini adalah untuk memetakan arsitektur sistem secara menyeluruh agar proses implementasi pada tahap simulasi dapat berjalan terarah dan meminimalisir kesalahan konfigurasi.

DOI: https://doi.org/10.31004/riggs.v5i2.12015

Dalam penelitian ini, perancangan dilakukan secara bertahap meliputi empat aspek utama. Dimulai dari Rancangan Jaringan untuk menentukan *topologi* interkoneksi antara pengguna, *firewall*, dan *server*; Rancangan Pengalamatan IP untuk mendefinisikan identitas logis perangkat dalam simulasi; Rancangan Sistem yang berfokus pada logika aturan (*rule logic*) *Mod Security* untuk memitigasi serangan; serta penentuan kebutuhan perangkat keras dan lunak sebagai syarat minimum lingkungan operasional. Keseluruhan rancangan ini menjadi landasan dasar dalam pembangunan prototipe sistem keamanan yang handal.

2.3. Simulation Prototyping

Simulation Prototyping adalah pengujian konsep dalam skala kecil atau lingkungan laboratorium. Sebelum diterapkan secara luas, rancangan tersebut diuji coba menggunakan mesin virtual atau server lokal untuk memastikan bahwa sistem yang diterapkan benar-benar berjalan tanpa mengganggu layanan yang lain.



Gambar 1. Tahapan Mode NDLC

3. Hasil dan Diskusi

Dalam penelitian ini akan ditampilkan hasil Analisis dan hasil skenario uji coba eksploitasi yang dilakukan oleh peneliti. Dibawah ini diitampilkan hasil analisis berupa pengumpulan data dari artikel terkait topik *SSRF* dan penggunaan *WAF* menggunakan metode studi literatur .

Tabel 1. Hasil Pengumpulan Data

Penulis	Tahun	Judul	Pembahasan
Yuchen Ji, Ting Dai, Zichao Zhou, Yutian Tang, Jingzhu He.	2025	*Artemis: Toward Accurate Detection of Server-Side Request Forgeries through LLM-Assisted Inter-procedural Path-Sensitive Taint Analysis*	Mengembangkan sistem deteksi bernama *Artemis* yang mempunyai serangkaian aturan untuk mencegah terjadinya eksploitasi kerentanan *SSRF*.

DOI: https://doi.org/10.31004/riggs.v5i2.12015

Muhammad Ridwan Fazli	2022	Analisis Kerentanan Insecure Design dan *Server Side Request Forgery (SSRF)* Dengan Metode *PTES* (Studi Kasus Objek *Buggy Web Application*)	Analisis dampak yang ditimbulkan dari kerentanan *SSRF* menggunakan metode *Penetration Testing Execution Standar (PTES)*.
Khadejah Al-talak, Dr. Onytra Abbass	2021	*Detecting Server-Side Request Forgery (SSRF) Attack by using Deep Learning Techniques*	Penerapan Tehnik *Deep Learning (LTSM Network)* untuk membuat *intelligent model* untuk mendeteksi serangan *SSRF*. Model yang di latih mendapatkan akurasi ~96.9%, menunjukkan kemampuan tinggi dalam mendeteksi pola serangan *SSRF*.
Enze Wang, Jianjun Chen, Wei Xie, Chuhan Wang, Yifei Gao, Zhenhua Wang, Haixin Duan, Yang Liu, Baosheng Wang	2024	*Where URLs Become Weapons: Automated Discovery of SSRF Vulnerabilities in Web Applications.*	Implementasi metode *SSRFuzz* , metode baru yang di gunakan untuk mengidentifikasi serangan kerentanan *SSRF*. Ditemukan 28 kerentanan *SSRF*, termasuk 25 yang sebelumnya tidak diketahui, dan 16 *CVE* baru dilaporkan dari 27 aplikasi nyata termasuk *Wordpress dan Joomla*.
Jacqueline Mukamisha, Aline Iradukunda, Elyse Manzi, Jema David Ndibwile	2025	*Mitigating Server-Side Request Forgery (SSRF) Attacks: An Empirical Analysis of Deep Learning-Based Approaches*	Pengembangan model Deep Learning , Long Short-Term Memory (LSTM) dan Bidirectional Encoder Representations from Transformers (BERT) menggunakan dataset URL CIC . Hasil yang ditunjukkan model LSTM melampaui BERT, mencapai akurasi 98,2%, dibandingkan dengan 94% milik BERT,yang Dimana mengungguli akurasi sebelumnya yaitu 96%

Analisis literatur menunjukkan penelitian kerentanan *SSRF* saat ini didominasi pendekatan deteksi, baik melalui alat otomatis maupun *deep learning*. Celah masih terdapat pada mitigasi preventif di lingkungan produksi, khususnya tingginya *false positive WAF* berbasis aturan seperti *ModSecurity* saat menangani logika bisnis kompleks. Skripsi ini mengisi celah tersebut dengan mengimplementasikan *ModSecurity* serta merancang *custom rules* yang ditala untuk menekan *false positive*, sehingga mampu memitigasi *SSRF* secara efektif tanpa mengorbankan fungsionalitas dan ketersediaan layanan bagi pengguna sah.

Selanjutnya ditampilkan hasil dari 2 skenario uji coba yang dilakukan oleh penguji yaitu , uji coba eksploitasi *SSRF* sebelum *ModSecurity* diaktifkan dan uji coba eksploitasi *SSRF* setelah *ModSecurity* diaktifkan.

Tabel 2. Hasil Uji Coba Eksploitasi SSRF Sebelum Mod Security Aktif

NO	UJI COBA	RESPONS			
		P1	P2	P3	P4
1	Eksploitasi *SSRF* Menggunakan Teknik *Common SSRF*	HTTP/2 200 OK	HTTP/2 200 OK	HTTP/2 200 OK	HTTP/2 200 OK
2	Eksploitasi *SSRF* Menggunakan Teknik *Bypass SSRF Filter*	HTTP/2 200 OK	HTTP/2 200 OK	HTTP/2 200 OK	HTTP/2 200 OK

Tabel 3 menampilkan empat *payload* (P1–P4) dengan target berbeda: alamat eksternal *ngrok*, halaman *router*, layanan *cockpit* server, dan layanan internal pada *port* 8002. Seluruh uji coba *Server-Side Request Forgery* (SSRF) sebelum penerapan ModSecurity menunjukkan aplikasi Katalog sangat rentan: *request* berbahaya

DOI: https://doi.org/10.31004/riggs.v5i2.12015

melalui parameter *stockApi* berhasil dieksploitasi. Keparahan meningkat karena adanya *Open Redirect* yang memungkinkan pemintasan filter logika aplikasi. Secara forensik, penyerang dapat mengendalikan *backend server* untuk melakukan *network mapping*, mengakses *router*, dan menembus layanan *port* internal 8002 yang seharusnya terisolasi.

Selanjut dilakukan pengujian terhadap *ModSecurity* dengan aturan *OWASP Core Rule Set* (CRS) dalam mod security mode *DetectionOnly*, yang hanya mencatat aktivitas mencurigakan ke dalam log tanpa memblokir *request* masuk. Tujuannya adalah mengevaluasi kemampuan aturan CRS dalam mendeteksi manipulasi *request* yang mengarah pada serangan *Server-Side Request Forgery* (SSRF).

Tabel 3. Uji Coba Eksploitasi SSRF Menggunakan Teknik Common SSRF dalam Mod Security Mode Detection Only

PAYLOAD	UJI COBA EKSPLOITASI SSRF MENGGUNAKAN TEKNIK COMMON SSRF
	Rule ID yang Terpicu
P1	-
P2	931100, *949110 & 980170*
P3	931100, *949110 & 980170*
P4	931100 dan *949110 & 980170*

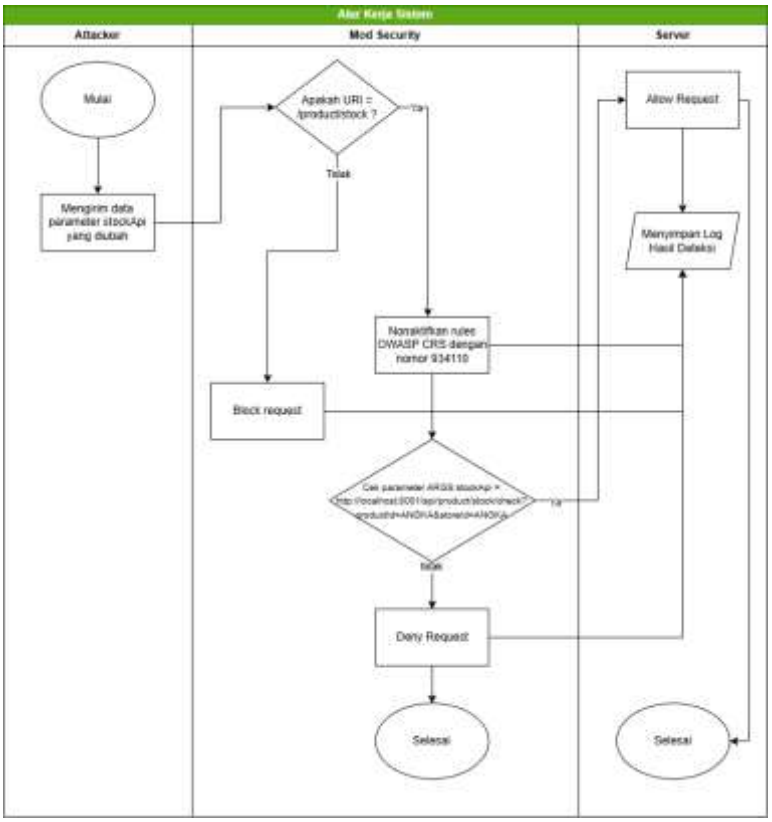
Berdasarkan Tabel 3, analisis deteksi *ModSecurity* terhadap *Common SSRF* menunjukkan tiga temuan utama: (1) *Blind spot* terjadi pada *payload* domain eksternal (*Ngrok*), di mana *rules* CRS tidak memicu peringatan karena tidak membatasi domain pihak ketiga dalam parameter URL; (2) *Rule ID 931100* berhasil mendeteksi target IP internal 192.168.1.1, meskipun diklasifikasikan sebagai *Remote File Inclusion* (RFI), membuktikan sensitivitas *regex* CRS terhadap alamat IP mentah; (3) *Rule ID 934110* secara konsisten mendeteksi kata kunci http://localhost pada berbagai variasi *port* tujuan (9090 dan 8002), menunjukkan deteksi mutlak pada level nama *host* lokal.

Tabel 4. Uji Coba Eksploitasi SSRF menggunakan Teknik Common SSRF Dalam Mod Security Mode Detection Only

NO	*PAYLOAD*	UJI COBA *EKSPLOITASI SSRF* MENGGUNAKAN *TEKNIK BYPASS SSRF FILTER*
		Rule ID yang Terpicu
1	P1	-
2	P2	931100, 949110, dan 980170
3	P3	931100 , *949110,* dan 980170
4	P4	920350, 931100, 94110, dan 980170

Berdasarkan Tabel 4, analisis *OWASP Core Rule Set* (CRS) terhadap teknik *bypass SSRF* via *Open Redirect* menunjukkan kelemahan signifikan. *Payload* yang disisipkan dalam parameter pengalihan (?path=) menuju domain eksternal (*Ngrok*) dan IP internal (192.168.1.1) berhasil lolos tanpa pemicuan *Rule ID* karena CRS hanya memeriksa permukaan parameter stockApi, mengabaikan *nested parameter*. Namun, *signature blacklist* agresif pada *Rule ID 934110* tetap mendeteksi alamat *loopback* (127.0.0.1) meski berada di akhir string. Seluruh percobaan menghasilkan status *200 OK* karena mode *DetectionOnly*, membuktikan tanpa pemblokiran aktif, serangan *SSRF* via *Open Redirect* tetap berhasil mengeksploitasi layanan internal target. Berdasarkan temuan ini, dilakukan penyesuaian *rules* untuk menutup celah tersebut.

DOI: https://doi.org/10.31004/riggs.v5i2.12015



Gambar 2. Rancangan Alur Kerja Custom Rule

Aturan ModSecurity ini memvalidasi *request* ke *endpoint* `/product/stock` dengan menonaktifkan rule bawaan *OWASP CRS ID* 934119 pada fase pertama untuk mencegah *false positive*, lalu menerapkan mekanisme *chain* pada fase 2 pada parameter *stockApi*. Hanya url `http://localhost:8001/api/product/stock/check?productId=<angka>&storeId=<angka>` yang diizinkan melalui *regex whitelist*; setiap ketidaksesuian langsung diblokir dengan 403 *Forbidden* dan pesan "*Invalid stockApi target*". Selanjutnya akan ditampilkan hasil uji coba eksploitasi setelah *modsecurity* aktif dan setelah diimplementasikan custom rule .

Tabel 5. Uji Coba Eksploitasi SSRF Setelah Mod Security Aktif

Penerapan *Custom Rules* pasca-fase *DetectionOnly* menunjukkan perubahan signifikan: seluruh delapan

UJI COBA	SETELAH *WAF* AKTIF			
	Rules ID Terpicu			
	P1	P2	P3	P4
Eksploitasi SSRF Menggunakan Teknik Common SSRF	60002 (Request Ditolak)	60002 (Request Ditolak)	60002 (Request Ditolak)	60002 (Request Ditolak)
Eksploitasi SSRF Menggunakan Teknik Bypass SSRF Filter	60002 (Request Ditolak)	60002 (Request Ditolak)	60002 (Request Ditolak)	60002 (Request Ditolak)

skenario *Common SSRF* maupun *Bypass SSRF Filter* kini diblokir total dengan status *403 Forbidden*,

DOI: https://doi.org/10.31004/riggs.v5i2.12015

membuktikan *WAF* pada mode aktif berhasil mengintervensi serangan sebelum mencapai *backend*. *Blind spot* sebelumnya—*payload* domain eksternal (*Ngrok*) dan IP internal via *Open Redirect*—tertutup rapat karena validasi berbasis *Allowlist* menolak setiap permintaan yang tidak sesuai spesifikasi URL yang diizinkan. Seluruh ancaman diidentifikasi seragam oleh *Rule ID 60002*, menggantikan aturan *signature-based* yang inkonsisten. Pendekatan *Positive Security Model* ini hanya mengizinkan koneksi ke *endpoint* stok sah (http://localhost:8001) sehingga memitigasi *SSRF* secara presisi tanpa *false positive* dan efektif menangkal *payload* tak terduga, termasuk variasi *Zero-Day*.

4. Kesimpulan

Berdasarkan hasil penelitian yang dilakukan , aplikasi berhasil diamankan dari upaya eksploitasi *SSRF* melalui penerapan *WAF* berbasis *ModSecurity* dengan kombinasi *OWASP CRS* dan *Custom Rules Whitelisting*. *Request* normal antar aplikasi yang terdapat di dalam *server* sebelumnya dideteksi sebagai serangan *SSRF* oleh aturan *OWASP CRS*. Setelah *custom rule* diterapkan, request antar aplikasi berhasil berjalan secara normal sekaligus diamankan dari upaya eksploitasi *SSRF*. Selain itu , aktifnya WAF berbasis modsecurity, mencatat semua aktivitas mencurigakan yang masuk kedalam aplikasi pada *server*. Dengan demikian penerapan *WAF* berbasis *ModSecurity* dalam mengatasi *Server Side Request Forgery* (SSRF) menjadi solusi yang efektif untuk mengamankan request antar aplikasi yang terdapat pada server.

Referensi

[1] E. Wang *et al.*, “Where URLs Become Weapons: Automated Discovery of SSRF Vulnerabilities in Web Applications.” [Online]. Available: https://github.com/SSRFuzz/SSRFuzz

[2] B. Zhang, C. Lou, Y. Lou, R. Ren, and Q. Wang, “SSRFinder: SSRF vulnerability detection and validation based on program dependency graphs and pre-trained models,” *Expert Syst. Appl.*, vol. 326, p. 132725, Sep. 2026, doi: 10.1016/J.ESWA.2026.132725.

[3] A. Abibulaiev, P. Pukach, and M. Vovk, “Context-Aware ML/NLP Pipeline for Real-Time Anomaly Detection and Risk Assessment in Cloud API Traffic,” *Mach. Learn. Knowl. Extr.*, vol. 8, no. 1, p. 25, Jan. 2026, doi: 10.3390/make8010025.

[4] Z. Qiu, S. Shao, Q. Zhao, and G. Jin, “Understanding and detecting server-side request races in web applications,” in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, New York, NY, USA: ACM, Aug. 2021, pp. 842–854. doi: 10.1145/3468264.3468594.

[5] N. Arora, P. Singh, S. Sahu, V. K. Keshari, and M. Vinoth Kumar, “Preventing SSRF (Server-Side Request Forgery) and CSRF (Cross-Site Request Forgery) Using Extended Visual Cryptography and QR Code,” 2021, pp. 215–227. doi: 10.1007/978-981-15-6707-0_20.

[6] B. Jabiyev, O. Mirzaei, A. Kharraz, and E. Kirda, “Preventing server-side request forgery attacks,” *Proceedings of the ACM Symposium on Applied Computing*, pp. 1626–1635, 2021, doi: 10.1145/3412841.3442036.

[7] Y. Ji, T. Dai, Z. Zhou, Y. Tang, and J. He, “Artemis: Toward Accurate Detection of Server-Side Request Forgeries through LLM-Assisted Inter-procedural Path-Sensitive Taint Analysis,” *Proceedings of the ACM on Programming Languages*, vol. 9, no. 1, Apr. 2025, doi: 10.1145/3720488.

[8] F. Fachri, “OPTIMASI KEAMANAN WEB SERVER TERHADAP SERANGAN BRUTE-FORCE MENGGUNAKAN PENETRATION TESTING,” vol. 10, no. 1, pp. 51–58, 2023, doi: 10.25126/jtiik.2023105872.

[9] Y. Zhou, E. Wang, and S. Ma, “SSRFSeek: An LLM-based Static Analysis Framework for Detecting SSRF Vulnerabilities in PHP Applications,” in *2025 IEEE 6th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, IEEE, Apr. 2025, pp. 939–944. doi: 10.1109/AINIT65432.2025.11035424.

[10] B. Dawadi, B. Adhikari, and D. Srivastava, “Deep Learning Technique-Enabled Web Application Firewall for the Detection of Web Attacks,” *Sensors*, vol. 23, no. 4, p. 2073, Feb. 2023, doi: 10.3390/s23042073.

[11] A. Thakkar and R. Lohiya, “A Review on Machine Learning and Deep Learning Perspectives of IDS for IoT: Recent Updates, Security Issues, and Challenges,” *Archives of Computational Methods in Engineering*, vol. 28, no. 4, pp. 3211–3243, Jun. 2021, doi: 10.1007/s11831-020-09496-0.

[12] M. R. Fazli, “ANALISIS KERENTANAN INSECURE DESIGN DAN SERVER SIDE REQUEST FORGERY (SSRF) DENGAN METODE PTES (STUDI KASUS OBJEK BUGGY WEB APPLICATION).”

[13] J. Mukamisha, A. Iradukunda, E. Manzi, and J. D. Ndibwile, “Mitigating Server-Side Request Forgery (SSRF) Attacks: An Empirical Analysis of Deep Learning-Based Approaches,” in *Proceedings - 2025 9th International Conference on Cryptography, Security and Privacy, CSP 2025*, 2025. doi: 10.1109/CSP66295.2025.00027.

[14] K. Al-Talak and O. Abbass, “Detecting Server-Side Request Forgery (SSRF) Attack by using Deep Learning Techniques.” [Online]. Available: www.ijacsa.thesai.org

[15] F. Naim, Rd. R. Saedudin, and S. K. Y. U. Hediyanto, “ANALYSIS OF WIRELESS AND CABLE NETWORK QUALITY-OF SERVICE PERFORMANCE AT TELKOM UNIVERSITY LANDMARK TOWER USING NETWORK DEVELOPMENT LIFE CYCLE (NDLC) METHOD ,” *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)* , vol. 7, no. 4, Dec. 2022.

DOI: https://doi.org/10.31004/riggs.v5i2.12015