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: 21301-21308
P-ISSN: 2963-9298, e-ISSN: 2963-914X

Penetration Testing White-Box dan Black-Box pada Web Server Menggunakan Large Language Model

Bayani Adam Sasaki[1], I Putu Hariyadi[2], Lilik Widyawati[3]
[1,2,3]Universitas Bumigora
[1]bayaniadamsasaki@gmail.com, [2]putu.hariyadi@universitasbumigora.ac.id, [3]lilikwidya@universitasbumigora.ac.id

Abstrak

Audit keamanan dan pengujian penetrasi (penetration testing) secara manual pada web server membutuhkan sumber daya yang besar, waktu yang lama, dan sangat bergantung pada keahlian individu sehingga berisiko menimbulkan temuan yang tidak konsisten. Di sisi lain, penggunaan Large Language Models (LLM) berbasis cloud untuk otomatisasi triage keamanan membuka risiko kebocoran data sensitif berupa konfigurasi sistem dan log kerentanan ke jaringan publik. Penelitian ini mengusulkan dan mengevaluasi kerangka kerja triage keamanan otomatis yang berjalan secara luring (offline) dan dipetakan ke standar Penetration Testing Execution Standard (PTES) menggunakan model lokal Qwen2:1.5b melalui framework Ollama. Alat berbasis Python yang dikembangkan, NULL Security System, memiliki 29 modul untuk audit internal (white-box) dan analisis lalu lintas aktif (black-box). Dalam lingkungan jaringan terisolasi VirtualBox, Kali Linux bertindak sebagai penyerang (menjalankan Nmap, Nuclei, dan FFUF) dan Ubuntu Server 24.04.4 LTS sebagai target. Hasil pengujian selama lima iterasi menunjukkan triage berbantuan LLM lokal mencapai konsistensi 100% dalam penilaian risiko dan ekstraksi Source IP. Rata-rata waktu pemrosesan (Time-to-Insight) tercatat 56 detik untuk triage white-box dan 1 menit 57 detik untuk triage black-box, merepresentasikan peningkatan efisiensi 5 hingga 10 kali lipat dibanding baseline manual. Sistem berhasil mengidentifikasi lalu lintas Nuclei secara konsisten melalui pemetaan pola perilaku URL (behavioral URL pattern) ketika metode pencocokan User-Agent tradisional gagal. Integrasi pemantau proses aktif (psutil) mencegah kelelahan CPU target dengan memicu analisis cadangan (fallback) berbasis kata kunci saat batas load average terlampaui, sehingga menjamin kelayakan sistem lokal dan kedaulatan data.

Kata Kunci: *LLM Lokal, PTES, Web Server, Penetration Testing*

1. Latar Belakang

Dalam era transformasi digital saat ini, *web server* merupakan infrastruktur utama penyedia layanan informasi yang rentan terhadap berbagai ancaman siber. Pengamanan *web server* membutuhkan proses audit secara berkala untuk mengidentifikasi celah keamanan baik dari sisi konfigurasi internal sistem (*white-box*) maupun ancaman serangan dari luar jaringan (*black-box*). Praktik audit keamanan tradisional atau *penetration testing* umumnya dilakukan secara manual mengikuti standar industri seperti *Penetration Testing Execution Standard* (PTES) [1], [2]. Namun, proses manual ini memiliki beberapa kelemahan signifikan, di antaranya membutuhkan waktu yang lama untuk menyisir berkas log yang sangat besar, sangat bergantung pada keahlian dan ketelitian individu penguji, serta rentan menghasilkan laporan yang kurang konsisten karena faktor kelelahan manusia [3], [4], [13].

Perkembangan teknologi Kecerdasan Buatan, khususnya *Large Language Models* (LLM), membuka peluang baru untuk membantu proses *security triage* yaitu penyaringan, analisis cepat, dan penentuan prioritas mitigasi atas temuan keamanan [5], [6], [15]. Meskipun demikian, penggunaan API LLM publik berbasis *cloud* (seperti OpenAI GPT atau Anthropic Claude) sangat dihindari dalam pengujian keamanan karena berisiko membocorkan informasi log sensitif, alamat IP, dan celah kerentanan *web server* ke server pihak ketiga di luar batas yurisdiksi organisasi [7], [8].

Untuk mengatasi tantangan tersebut, penelitian ini mengusulkan pengembangan kerangka kerja *penetration testing* berbantuan *Local LLM* (LLM yang dijalankan secara luring penuh) untuk otomatisasi *security triage* pada *web server* uji. Sistem dijalankan menggunakan model Qwen2:1.5b di bawah *framework* Ollama pada mesin *host* Windows yang terhubung secara luring via jaringan *Host-only Adapter* ke mesin target Ubuntu Server 24.04.4 LTS [9], [10]. Audit *white-box* dan *black-box* dijalankan oleh skrip Python bernama NULL Security System yang dikembangkan dengan 29 modul pemeriksaan konfigurasi internal dan pemantauan lalu lintas aktif.

Penelitian ini memetakan seluruh aktivitas uji secara ketat ke dalam tahapan-tahapan PTES (dari *pre-engagement* hingga *reporting*) dan melakukan evaluasi komparatif secara *head-to-head* antara efisiensi *triage* berbantuan LLM

lokal terhadap *baseline* analisis manual tradisional. Kontribusi utama penelitian ini adalah membuktikan kelayakan operasional *offline LLM* untuk *triage* keamanan berkinerja tinggi, menjamin kedaulatan data sensitif, mengimplementasikan mekanisme *resource-guarding fallback* untuk mencegah kelelahan CPU pada target, serta mendeteksi pemindaian terselubung (seperti Nuclei) secara perilaku (*behavioral URL pattern*) ketika tanda *User-Agent* telah disamarkan [14].

2. Metode Penelitian

2.1. Metodologi Pengujian Penetrasi (PTES)

Penelitian ini mengadopsi ketujuh fase dalam kerangka *Penetration Testing Execution Standard* (PTES) yang diformulasikan secara teknis di laboratorium terisolasi [2], [4], [12].

1. Pre-engagement Interactions: Menetapkan batas pengujian legal pada jaringan terisolasi VirtualBox untuk memastikan lalu lintas uji siber tidak bocor ke internet publik.
2. Intelligence Gathering: Memetakan konfigurasi dasar target, melakukan *banner grabbing*, dan memvalidasi keterbukaan *port* layanan (*port* 80 untuk HTTP, *port* 22 untuk SSH).
3. Threat Modeling: Menilai vektor ancaman dari luar (eksploitasi lalu lintas pemindaian) dan dari dalam (audit konfigurasi berkas sistem, hak akses, dan otentikasi).
4. Vulnerability Analysis: Program NULL Security System mengekstrak konfigurasi server dan rekam jejak log Nginx untuk disaring menjadi artefak temuan kerentanan awal.
5. Exploitation: Mensimulasikan serangan pemindaian dan kerentanan aktif menggunakan *attacker tools* (Nmap, Nuclei, FFUF) dari mesin Kali Linux untuk membangkitkan log logis pada target.
6. Post-Exploitation: Membedah memori proses aktif target menggunakan modul psutil di Python untuk mencari sisa proses berbahaya dan memulihkan kondisi server ke kondisi dasar (*base snapshot*).
7. Reporting: Mengirimkan data mentah *triage* ke LLM lokal untuk dikonversi menjadi laporan ringkas terstruktur (Status, Masalah, Tindakan Mitigasi) yang dikirimkan secara otomatis via Telegram Bot API kepada administrator jaringan [11].

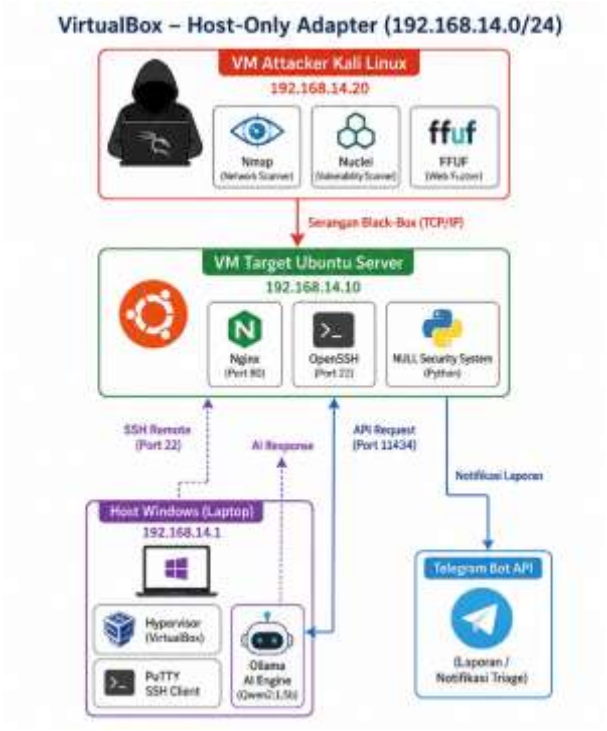
2.2. Rancangan Lingkungan Jaringan Laboratorium

Uji coba dirancang menggunakan topologi jaringan terisolasi di dalam *hypervisor* VirtualBox dengan *subnet* `192.168.14.0/24`. Topologi ini terdiri atas tiga entitas utama yang saling terhubung.

1. Host Windows (IP: 192.168.14.1): Mesin fisik yang menjalankan layanan *Local LLM* Ollama dengan model Qwen2:1.5b.
2. VM Attacker - Kali Linux (IP: 192.168.14.20): Bertindak sebagai peretas eksternal yang melepaskan pemindaian aktif.
3. VM Target - Ubuntu Server 24.04.4 LTS (IP: 192.168.14.10): Menjalankan *web server* Nginx dan mengeksekusi skrip NULL Security System berbasis Python.

Rancangan lingkungan uji coba dan topologi jaringan yang digunakan disajikan pada gambar 1 berikut.

DOI: https://doi.org/10.31004/riggs.v5i2.11841



Gambar 1. Rancangan Lingkungan Uji Coba pada VirtualBox

2.3. Spesifikasi Perangkat Keras dan Perangkat Lunak

Sistem dijalankan pada spesifikasi perangkat keras dan lunak dengan rincian sebagai berikut.

1. Perangkat Keras: Laptop ASUS TUF Gaming F15 dengan prosesor Intel Core i5 (12 *cores*, VT-x aktif), RAM 16 GB DDR4, dan GPU NVIDIA GeForce RTX 3050.
2. Perangkat Lunak: VirtualBox 7.0, Kali Linux 2024.1, Ubuntu Server 24.04.4 LTS, Nginx 1.24.0, Python 3.11+, Ollama 0.1.48, dan Telegram Desktop.

2.4. Perancangan Skrip Sensor Python (*NULL Security System*)

Skrip Python dikembangkan dengan 29 modul modular yang diklasifikasikan ke dalam 15 pilihan menu CLI utama. Tiga fungsi penting dalam kode program dirancang dengan alur sebagai berikut.

2.4.1 Pemindaian Port TCP Berbasis Socket

Menguji keterbukaan *port* menggunakan pustaka bawaan socket secara cepat untuk validasi awal. Cuplikan kode pemindaian ditunjukkan pada program 1 berikut.

```
def capture_check_port_output(host, port):
    captured_output = []
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.settimeout(1)
    result = sock.connect_ex((host, port))
    if result == 0:
        captured_output.append(f"[+] Port {port} di {host} TERBUKA.")
        try:
            service = socket.getservbyport(port, "tcp")
            captured_output.append(f"    Layanan: {service}")
        except OSError:
            captured_output.append("    Layanan: Tidak diketahui")
    sock.close()
    return "\n".join(captured_output), (result == 0)
```

2.4.2 Log Monitoring Real-Time & Ekstraksi IP

Membaca berkas log server yaitu `/var/log/nginx/access.log` (log akses Nginx), `/var/log/nginx/error.log` (log kesalahan Nginx), dan `/var/log/auth.log` (log otentikasi SSH)

DOI: https://doi.org/10.31004/riggs.v5i2.11841

secara *real-time* menggunakan teknik *tailing* dan ekspresi reguler (*regex*) untuk mengekstrak alamat IP penyerang.

2.4.3 Komunikasi API Ollama dan Fallback Average Load

Fungsi interaksi luring mengirimkan data log terkompresi ke API Ollama pada *host* Windows. Jika beban CPU target (diperiksa melalui `os.getloadavg()`) melebihi batas `AI_MAX_LOAD` (1.8), sistem secara otomatis mengalihkan proses *triage* ke *Fallback Analysis* berbasis pencocokan kata kunci lokal untuk melindungi stabilitas VM target.

Untuk memahami alur kerja sistem secara keseluruhan dari pemindaian serangan hingga notifikasi Telegram terkirim, dirancang sebuah diagram alur kerja (*flowchart*) yang ditunjukkan pada gambar 2 berikut.

Gambar 2. Rancangan Alur Kerja Pengujian dan Triage LLM

3. Hasil dan Pembahasan

3.1. Hasil Implementasi Audit *White-Box*

Audit *white-box* dilakukan dengan mengeksekusi 29 modul skrip pemeriksaan keamanan internal pada mesin Ubuntu Server target. Modul audit mencakup pemeriksaan layanan SSH, konfigurasi hak akses berkas direktori penting, parameter pengetatan *kernel* (*kernel hardening*), *port* aktif, dan aturan *firewall* UFW. Detail pembagian modul dan hasil temuan disajikan pada tabel 1 berikut.

DOI: https://doi.org/10.31004/riggs.v5i2.11841

Tabel 1. Hasil Audit Modul White-Box pada NULL Security System

No	Nama Modul	Parameter Pemeriksaan	Status Kerentanan	Rekomendasi Mitigasi
1	ssh_checker.py	Otentikasi kata sandi & akses *root*	HIGH	Matikan *PasswordAuthentication* & *PermitRootLogin*
2	user_group_checker.py	Akun dengan UID 0 (setara *root*)	LOW	Hanya akun root yang diizinkan memiliki UID 0
3	network_config_checker.py	Antarmuka jaringan & status IP	LOW	Isolasi antarmuka yang tidak terpakai
4	port_scanner.py	Port terbuka tidak terpakai	MEDIUM	Tutup *port* 21 (FTP) jika tidak digunakan
5	system_hardening_checker.py	Konfigurasi sysctl *kernel*	MEDIUM	Aktifkan perlindungan *IP Spoofing* & *ICMP Redirects*
6	auditd_checker.py	Keaktifan Audit Daemon (auditd)	HIGH	Pasang dan aktifkan paket auditd untuk log forensik

3.2. Hasil Implementasi Deteksi *Black-Box*

Deteksi *black-box* bertujuan untuk menguji keandalan sistem dalam mengidentifikasi lalu lintas serangan yang dilepaskan oleh Kali Linux secara *real-time*. Hasil pengujian disajikan pada tabel 2 berikut.

Tabel 2. Hasil Deteksi Serangan Black-Box pada NULL Security System

Nama Alat Peretas	Indikator Deteksi Utama	Akurasi Ekstraksi IP	Metode Identifikasi Sistem	Status Risiko AI
Nmap	*User-Agent*: Nmap Scripting Engine	100% (192.168.14.20)	Pencocokan *string User-Agent*	**LOW**
FFUF	*User-Agent*: ffuf	100% (192.168.14.20)	Pencocokan *string User-Agent*	**LOW**
Nuclei	Pola URL: /wp-content/, /cgi-bin/	100% (192.168.14.20)	*Behavioral URL Pattern Matching*	**LOW**

Salah satu temuan terpenting dalam penelitian ini adalah pendeteksian alat pemindai Nuclei. Nuclei secara bawaan memanipulasi *User-Agent* miliknya untuk menyamar sebagai *browser* normal sehingga analisis log tradisional berbasis pencarian kata kunci *User-Agent* (seperti `grep -i "nuclei"`) akan selalu gagal mendeteksinya. Namun, NULL Security System berhasil mendeteksi serangan Nuclei secara akurat dan konsisten berdasarkan perilaku permintaan URL (*behavioral URL pattern*) yang khas, seperti pengaksesan direktori sensitif `/cgi-bin/`, `/wp-content/plugins/`, dan *file* `/elfinder.html` dalam rentang waktu yang sangat rapat.

DOI: https://doi.org/10.31004/riggs.v5i2.11841

3.3. Hasil Integrasi Notifikasi Telegram

Saat temuan *triage* selesai dianalisis oleh LLM Qwen2:1.5b, data dikonversi secara dinamis menjadi JSON laporan keamanan dan ditransmisikan ke API Telegram. Contoh tampilan notifikasi yang diterima oleh administrator jaringan disajikan pada gambar 3 berikut.

Gambar 3. Representasi Teks Laporan Triage yang Diterima via Telegram

3.4. Analisis Evaluasi Iteratif

Untuk menguji keandalan dan konsistensi sistem, dilakukan pengujian berulang sebanyak 5 kali iterasi untuk masing-masing skenario pengujian. Pengukuran waktu (*Time-to-Insight*) dicatat secara presisi menggunakan stempel waktu (*timestamp*) sistem Unix melalui penanda awal dan akhir eksekusi perintah terminal.

3.4.1 Hasil Pengujian Skenario 1 (*White-Box Triage* - 5 Iterasi)

Hasil pengukuran durasi pemrosesan dan konsistensi penilaian pada audit internal disajikan pada tabel 3 berikut.

Tabel 3. Data Hasil Pengujian Skenario 1 (White-Box Triage)

Iterasi	Durasi Pengujian (detik)	Jumlah Pemeriksaan	Jumlah Notifikasi Telegram	Status Risiko AI	Konsistensi Hasil (%)
1	56	6	6	LOW	100%
2	54	6	6	LOW	100%
3	58	6	6	LOW	100%
4	55	6	6	LOW	100%
5	57	6	6	LOW	100%
Rata-rata	56 detik				

3.4.2 Hasil Pengujian Skenario 1 (*White-Box Triage* - 5 Iterasi)

Hasil deteksi serangan luar (Nmap, Nuclei, FFUF) disajikan pada tabel 4 berikut.

DOI: https://doi.org/10.31004/riggs.v5i2.11841

Tabel 4. Data Hasil Pengujian Skenario 2 (Black-Box Triage)

Iterasi	Durasi Pengujian (detik)	Events Nmap	Events Nuclei	Events FFUF	Source IP Terdeteksi	Status Risiko AI
1	117	4	6	100	192.168.14.20	LOW
2	115	4	6	100	192.168.14.20	LOW
3	122	4	6	100	192.168.14.20	LOW
4	118	4	6	100	192.168.14.20	LOW
5	114	4	6	100	192.168.14.20	LOW
Rata-rata	117.2 detik					

3.5. Pengujian Komparatif Terhadap *Baseline* Manual

Untuk mengukur efektivitas dan efisiensi sistem yang diusulkan, dilakukan perbandingan performa *head-to-head* terhadap *baseline* manual tradisional yang merujuk pada standar industri keamanan siber. Hasil rekapitulasi komparatif disajikan pada tabel 5 berikut.

Tabel 5. Rekapitulasi Perbandingan Hasil Uji vs Baseline Manual

Metrik Evaluasi	Baseline Manual Tradisional	Sistem Triage Berbantuan LLM Lokal
Rata-rata Durasi Triage	WB: 5 - 15 Menit	WB: 56 Detik
	BB: 10 - 20 Menit	BB: 1 Menit 57 Detik
Penyusunan Laporan	Manual (Menyalin log ke dokumen teks)	Otomatis terkirim ke Telegram
Keandalan Deteksi Nuclei	Rendah (Gagal jika *User-Agent* disamarkan)	Tinggi (Berhasil via *Behavioral URL Pattern*)
Ekstraksi Alamat IP Penyerang	Manual (Menggunakan perintah awk/grep)	Otomatis diekstrak secara instan
Konsistensi Penilaian Risiko	Fluktuatif (Dipengaruhi subjektivitas manusia)	100% Konsisten (Berdasarkan parameter deterministik)

Berdasarkan data tabel 5 berikut, sistem *triage* berbantuan LLM lokal terbukti secara signifikan mengungguli performa audit manual tradisional pada seluruh metrik evaluasi. Pada domain *white-box*, sistem memangkas waktu audit menjadi rata-rata 56 detik (meningkat 5 hingga 15 kali lebih cepat). Pada domain *black-box*, sistem mampu menyaring dan menggolongkan tiga serangan siber sekaligus dalam waktu 1 menit 57 detik (meningkat 5 hingga 10 kali lebih cepat). Hal ini memberikan bukti kuat bahwa pemanfaatan *offline* LLM sebagai asisten *triage* dapat mengefisiensikan operasi tim pertahanan siber (*blue team*) tanpa mengekspos kerahasiaan data *server* ke internet publik.

4. Kesimpulan

Penelitian ini telah berhasil merancang, mengimplementasikan, dan mengevaluasi kerangka kerja *triage* keamanan otomatis berbasis Python (NULL Security System) berbantuan model bahasa besar lokal (Qwen2:1.5b via Ollama) yang diselaraskan dengan standar PTES. Penggunaan model lokal yang ditempatkan pada *host* Windows terisolasi via jaringan *Host-only Adapter* terbukti efektif menjaga privasi log keamanan dan kedaulatan data sensitif *web server*. Pengujian empiris sebanyak 5 kali iterasi membuktikan keandalan sistem dengan efisiensi waktu pemrosesan (*Time-to-Insight*) mencapai rata-rata 56 detik untuk audit internal (*white-box*) dan 1 menit 57 detik untuk deteksi eksternal (*black-box*), menunjukkan akselerasi kinerja sebesar 5 hingga 10 kali lipat dibanding

DOI: https://doi.org/10.31004/riggs.v5i2.11841

metode analisis manual tradisional. Lebih jauh lagi, sistem terbukti andal dalam mendeteksi serangan pemindaian Nuclei secara perilaku (*behavioral URL pattern*) dan aman dari ancaman kelelahan memori server berkat implementasi mekanisme pemantauan proses aktif (`psutil`) yang memicu *fallback* otomatis secara dinamis.

Referensi

[1] Yosua Ade Pohan, Yuhandri Yunus, and Sumijan, "Meningkatkan Keamanan Webserver Aplikasi Pelaporan Pajak Daerah Menggunakan Metode Penetration Testing Execution Standar," Jurnal Sistim Informasi dan Teknologi, pp. 1–6, Mar. 2021, doi: 10.37034/jsisfotek.v3i1.36.

[2] H. Zhang, J. Wang, Y. Wang, M. Li, J. Song, and Z. Liu, "ICVTest: A Practical Black-Box Penetration Testing Framework for Evaluating Cybersecurity of Intelligent Connected Vehicles," Applied Sciences (Switzerland), vol. 14, no. 1, Jan. 2024, doi: 10.3390/app14010204.

[3] F. Akmalia Maylani, M. Tahir, N. Nataswa Juniar, D. Sari, and W. Ayu Zulaica, "Analisis Keamanan Website E-Library Kampus dengan Metode PTES (Penetration Testing Execution Standard)," 2025.

[4] Hasibuan Marzuki and Elhanafi Andi Marwan, "Penetration Testing Sistem Jaringan Komputer Menggunakan Kali Linux untuk Mengetahui Kerentanan Keamanan Server dengan Metode Black Box," sudo Jurnal Teknik Informatika, vol. 1, no. 4, pp. 171–177, Dec. 2022, doi: 10.56211/sudo.v1i4.160.

[5] K. U. Sarker, F. Yunus, and A. Deraman, "Penetration Taxonomy: A Systematic Review on the Penetration Process, Framework, Standards, Tools, and Scoring Methods," Jul. 01, 2023, Multidisciplinary Digital Publishing Institute (MDPI). doi: 10.3390/su151310471.

[6] E. Hilario, S. Azam, J. Sundaram, K. Imran Mohammed, and B. Shanmugam, "Generative AI for pentesting: the good, the bad, the ugly," Int. J. Inf. Secur., vol. 23, no. 3, pp. 2075–2097, Jun. 2024, doi: 10.1007/s10207-024-00835-x.

[7] N. O. Jaffal, M. Alkhanafseh, and D. Mohaisen, "Large Language Models in Cybersecurity: A Survey of Applications, Vulnerabilities, and Defense Techniques," Sep. 01, 2025, Multidisciplinary Digital Publishing Institute (MDPI). doi: 10.3390/ai6090216.

[8] S. Ćirković, V. Mladenović, S. Tomić, D. Drljača, and O. Ristić, "Utilizing Fine-Tuning of Large Language Models for Generating Synthetic Payloads: Enhancing Web Application Cybersecurity through Innovative Penetration Testing Techniques," Computers, Materials and Continua, vol. 82, no. 3, pp. 4409–4430, 2025, doi: 10.32604/cmc.2025.059696.

[9] C. Huang, X. Huang, N. P. Tran, and A. M. Fard, "Model Context Protocol Threat Modeling and Analysis of Vulnerabilities to Prompt Injection with Tool Poisoning," Journal of Cybersecurity and Privacy, vol. 6, no. 3, p. 84, May 2026, doi: 10.3390/jcp6030084.

[10] K. Lee, S. Yang, J. Jeong, Y. Lee, and D. Shin, "Enhancing Security and Applicability of Local LLM-Based Document Retrieval Systems in Smart Grid Isolated Environments," Electronics (Switzerland), vol. 14, no. 17, Sep. 2025, doi: 10.3390/electronics14173407.

[11] H. Liu and J. Xu, "Empirical Analysis of Internal Hallucination Detection in Quantized LLMs: Layer Dynamics and White-Box Benchmarks," Electronics (Basel)., vol. 15, no. 9, p. 1802, Apr. 2026, doi: 10.3390/electronics15091802.

[12] N. S. Mathews, Y. Brus, Y. Aafer, M. Nagappan, and S. McIntosh, "LLbezpeky: Leveraging Large Language Models for Vulnerability Detection," Feb. 2024, [Online]. Available: http://arxiv.org/abs/2401.01269

[13] M. Amirul, N. Shamsudin, and M. F. Zolkipli, "A Comparative Analysis of Penetration Testing Tools for Network Vulnerability Assessment," 2025. [Online]. Available: www.majmuah.com

[14] L. A. Pfuño Alccahuamani, A. Meza Bautista, and H. Rojas, "Hybrid Web Architecture with AI and Mobile Notifications to Optimize Incident Management in the Public Sector," Computers, vol. 15, no. 1, p. 47, Jan. 2026, doi: 10.3390/computers15010047.

[15] M. Ridho, A. Hafizh, I. Dani, and T. Ariyadi, "Peningkatan Keamanan SSH Server Berbasis Linux melalui Implementasi Fail2Ban dan Uji Serangan Brute Force," Jurnal Penelitian Multidisiplin Bangsa, vol. 1, no. 12, 2025, [Online]. Available: https://ejournal.amirulbangunbangsapublishing.com/index.php/jpnmb/index

[16] D. Moreira, J. P. Seara, J. P. Pavia, and C. Serrão, "Intelligent Platform for Automating Vulnerability Detection in Web Applications," Electronics (Switzerland), vol. 14, no. 1, Jan. 2025, doi: 10.3390/electronics14010079.

DOI: https://doi.org/10.31004/riggs.v5i2.11841