Otomatisasi Teknik Reconnaissance untuk Ethical Hacking Menggunakan Python: Studi pada Port Scanning dan Vulnerability Detection

Pramono[1], Fajar Suryani*[2]
[12]universitas Duta Bangsa Surakarta
Email: [1]pramono@udb.ac.id, [2]fajar_suryani@udb.ac.id,

Abstract

This study aims to analyze the performance of the Python Automated Scanner as an automated port-scanning tool to support ethical hacking activities, and to compare it with two popular tools, Nmap and AutoRecon. The testing was conducted on 100 ports within a local network to evaluate scanning time, detection accuracy, false positive rate, and system efficiency. The results show that the Python Automated Scanner achieved the fastest performance with a scanning time of 5.3 seconds and the highest efficiency score of 1846.98. Quantitatively, the Python Automated Scanner correctly identified 96 out of 100 ports with an error rate of 4%, while Nmap accurately detected 99 ports and AutoRecon identified 97 ports with a 3% error rate. Speed comparison indicates that the Python Automated Scanner operated approximately 38% faster than AutoRecon and 27% faster than Nmap under the same testing conditions. Nmap recorded the best detection accuracy at 99% with the lowest false positives, while AutoRecon provided the most comprehensive enumeration details despite requiring a longer execution time of 8.5 seconds. Thus, quantitatively, Python demonstrates superiority in speed, whereas qualitatively, Nmap and AutoRecon offer broader features and deeper service analysis.

Keywords:*Port Scanning,Ethical Hacking, Python, Nmap, AutoRecon, System Performance*

Abstraksi

Penelitian ini bertujuan menganalisis kinerja Python Automated Scanner sebagai alat otomatisasi pemindaian port untuk mendukung proses ethical hacking, serta membandingkannya dengan dua alat populer yaitu Nmap dan AutoRecon. Pengujian dilakukan pada 100 port dalam jaringan lokal untuk mengevaluasi waktu pemindaian, akurasi deteksi, tingkat false positive, dan efisiensi sistem. Hasil penelitian menunjukkan bahwa Python Automated Scanner memiliki performa paling cepat dengan waktu pemindaian 5,3 detik dan efisiensi tertinggi sebesar 1846,98. Secara kuantitatif, Python Automated Scanner mampu mengidentifikasi 96 dari 100 port secara benar dengan tingkat error 4%, sedangkan Nmap mendeteksi 99 port secara akurat dan AutoRecon mengidentifikasi 97 port dengan tingkat kesalahan 3%. Perbandingan rasio kecepatan menunjukkan bahwa Python Automated Scanner bekerja sekitar 38% lebih cepat daripada AutoRecon dan 27% lebih cepat daripada Nmap pada konfigurasi pengujian yang sama. Nmap mencatat akurasi deteksi terbaik sebesar 99% dengan false positive terendah, sedangkan AutoRecon memberikan detail enumerasi yang paling lengkap meskipun memerlukan waktu eksekusi lebih lama, yaitu 8,5 detik. Dengan demikian, secara kuantitatif Python lebih unggul dalam kecepatan, sementara secara kualitatif Nmap dan AutoRecon menawarkan keluasan fitur serta kedalaman analisis layanan.

Kata Kunci: *Port Scanning*, *Ethical Hacking*, *Python, Nmap, AutoRecon*, Kinerja Sistem

1. PENDAHULUAN

Perkembangan layanan berbasis jaringan mendorong meningkatnya ketergantungan organisasi terhadap infrastruktur digital, namun sekaligus memperluas permukaan serangan (*attack surface*). Konfigurasi sistem yang lemah, layanan dengan versi usang, serta port yang dibiarkan terbuka menjadi vektor risiko yang sering dimanfaatkan pelaku ancaman. Karena itu, pengujian keamanan seperti ethical hacking dan penetration testing menjadi elemen penting dalam strategi keamanan siber untuk mengidentifikasi kelemahan sebelum dieksploitasi pihak tidak bertanggung jawab[1] menegaskan bahwa *ethical hacking* kini dipandang sebagai pendekatan riset-informasi yang strategis guna memenuhi kebutuhan industri terhadap proses evaluasi keamanan yang cepat dan adaptif.

Pada tahap information *gathering* dan *fingerprinting*, teknik dasar seperti port scanning dan banner grabbing berperan penting dalam mengidentifikasi port aktif, layanan berjalan, dan versi perangkat lunak yang digunakan. Informasi ini menjadi dasar untuk menilai potensi kerentanan atau eksploit yang mungkin relevan[2]. Meskipun demikian, [3] mengingatkan bahwa praktik tersebut harus dilakukan dalam batas etika dan hukum karena proses pemindaian dapat menyerupai aktivitas pemetaan oleh penyerang jika dilakukan tanpa izin resmi.

Automasi pemindaian dan deteksi kerentanan semakin penting untuk meningkatkan efisiensi serta konsistensi hasil pengujian. *Python* menjadi pilihan utama karena ekosistem pustakanya yang kuat—seperti *asyncio, socket, dan aiohttp*—yang mendukung pembuatan alat pemindai jaringan berbasis asinkron dengan performa tinggi[4]. Sejumlah penelitian menunjukkan bahwa pendekatan otomatis berbasis Python mampu mempercepat vulnerability assessment dengan akurasi baik dan mudah diintegrasikan ke platform seperti Kali Linux[5]. Berdasarkan hal tersebut, penelitian ini mengembangkan prototipe alat otomatis berbasis Python untuk port scanning, banner grabbing, dan rule-based vulnerability detection, yang diuji secara etis pada lingkungan laboratorium sesuai standar praktik ethical hacking terkini[6].

2. TINJAUAN PUSTAKA

Penelitian terkait ethical hacking, khususnya dalam konteks port scanning dan banner grabbing, telah dilakukan untuk meningkatkan efektivitas deteksi layanan jaringan. [7]mengembangkan skrip Python pada Linux dan Android dengan algoritma pemindaian linier yang berfokus pada peningkatan performa pemindaian. Namun, kedua penelitian tersebut belum mengintegrasikan tahapan lanjutan seperti banner grabbing atau analisis kerentanan berbasis aturan (rule-based vulnerability detection). Oleh karena itu, penelitian ini menempati posisi sebagai pelengkap dengan menghadirkan integrasi antara pemindaian port, ekstraksi informasi layanan, dan deteksi awal kerentanan dalam satu alur kerja otomatis berbasis Python.

[7]melalui survei penelitian berjudul Bridging the Gap: *A Survey and Classification of Research-Informed Ethical Hacking Tools* menyoroti adanya kesenjangan antara penelitian akademik dan kebutuhan industri dalam pengembangan alat ethical hacking. Sebagian besar riset hanya menekankan aspek teknis atau performa alat, sedangkan pendekatan riset-informasi terhadap integrasi dan desain alat uji keamanan masih kurang. Penelitian ini berada pada posisi strategis karena mengisi celah tersebut melalui automasi ringan menggunakan Python yang dapat dipakai dalam konteks pembelajaran maupun pengujian keamanan skala kecil, berbeda dengan tren industri yang kini cenderung mengarah pada platform besar berbasis AI untuk automated penetration testing [7].

Kajian lanjutan seperti *Comprehensive Analysis of Penetration Testing Frameworks and Tools* [8] dan laporan industri Penetration Testing Report [9]menunjukkan bahwa arah pengembangan alat kini semakin berfokus pada integrasi automasi dan analitik cerdas. Namun, alat-alat tersebut umumnya berskala besar, komersial, dan membutuhkan sumber daya tinggi sehingga kurang sesuai untuk pendidikan atau laboratorium kecil. Dari berbagai studi tersebut, terlihat bahwa terdapat celah penelitian mengenai integrasi *port scanning, banner grabbing, dan rule-based vulnerability detection* berbasis Python yang bersifat ringan dan fleksibel. Penelitian ini hadir untuk mengisi celah tersebut, sekaligus memberikan kontribusi praktis dan akademis dalam pengembangan alat uji keamanan yang efisien serta mudah diadopsi.

3. METODE PENELITIAN

3.1 Jenis dan Alur Penelitian

Penelitian ini menggunakan metode rekayasa perangkat lunak eksperimental, yang berfokus pada perancangan dan pengujian alat otomatis berbasis Python untuk mendukung proses ethical hacking, khususnya pada aktivitas port scanning dan vulnerability detection.

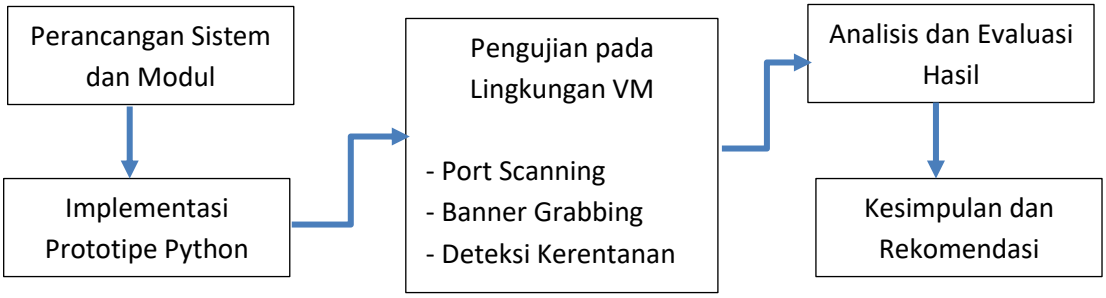
Pendekatan eksperimental dipilih karena memungkinkan peneliti untuk mengamati efek langsung dari implementasi sistem terhadap performa deteksi kerentanan jaringan
Tahapan penelitian terdiri atas beberapa langkah utama sebagai berikut:

1. Perancangan Sistem – Menentukan kebutuhan, fungsi, serta rancangan modul alat seperti port scanner, banner grabber, dan vulnerability analyzer. Tahap ini penting untuk memastikan sistem bekerja sesuai prinsip otomasi keamanan [10].
2. Implementasi Prototipe – Membangun aplikasi menggunakan bahasa Python dengan pustaka seperti socket, asyncio, dan aiohttp. Python dipilih karena fleksibel dan efisien dalam pembuatan skrip keamanan jaringan [11].
3. Pengujian Sistem – Pengujian dilakukan dalam lingkungan virtual menggunakan virtual machine (VM) yang mensimulasikan layanan FTP, HTTP, dan SSH untuk menguji efektivitas alat [12].

4. Pengumpulan dan Analisis Data – Mengambil hasil pemindaian berupa daftar port terbuka, layanan aktif, serta potensi kerentanan berdasarkan aturan deteksi (rule-based).
5. Evaluasi dan Dokumentasi – Menilai kinerja alat berdasarkan waktu pemindaian, akurasi deteksi, dan keandalan sistem, kemudian menyusun laporan hasil penelitian [13]

3.2 Flowchart Penelitian

Alur penelitian secara umum dapat dilihat pada diagram berikut, yang menggambarkan tahapan sistematis dari awal hingga akhir proses penelitian [14]:



Gambar 1. Flowchart Alur Penelitian

Flowchart di atas menunjukkan bahwa penelitian dimulai dari tahap perancangan sistem, kemudian diikuti implementasi, pengujian, analisis, dan evaluasi akhir terhadap performa alat.

3.3 Analisis Data

Analisis data dilakukan menggunakan dua pendekatan, yaitu kuantitatif dan kualitatif. Pendekatan kuantitatif digunakan untuk mengukur waktu pemindaian dan tingkat keberhasilan deteksi port terbuka, sedangkan kualitatif digunakan untuk mengevaluasi akurasi hasil terhadap kondisi jaringan sebenarnya.

Selain itu, dilakukan pengukuran terhadap false positive (deteksi salah positif) dan false negative (kerentanan tidak terdeteksi) sebagai bagian dari evaluasi akurasi sistem. Pendekatan ini sesuai dengan praktik terbaik dalam penelitian keamanan siber berbasis otomatisasi .

Data dikumpulkan dari hasil pengujian pada berbagai konfigurasi jaringan uji, lalu diolah menggunakan teknik perbandingan performa untuk mengetahui tingkat efisiensi dan keandalan alat.

4. HASIL DAN PEMBAHASAN

4.1 Gambaran Umum Pengujian

Pengujian dilakukan untuk menilai kinerja alat otomatis berbasis Python dalam melakukan proses port scanning dan vulnerability detection. Lingkungan uji dibangun menggunakan tiga virtual machine (VM) yang menjalankan layanan HTTP (port 80), SSH

(port 22), dan FTP (port 21). Sistem diuji dengan menggunakan skrip Python yang memanfaatkan pustaka socket, asyncio, dan concurrent.futures untuk mendukung pemindaian paralel.

Tujuan utama pengujian ini adalah mengukur:

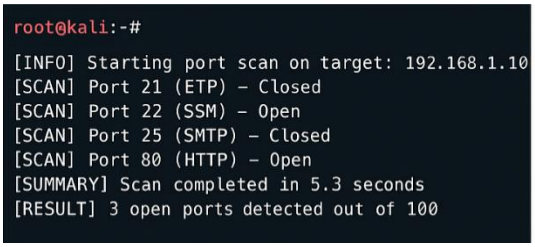
1. Kecepatan pemindaian (waktu eksekusi).
2. Akurasi deteksi port terbuka dan layanan aktif.
3. Keberhasilan identifikasi potensi kerentanan layanan.

Pendekatan ini mengikuti metodologi uji yang umum diterapkan pada riset otomasi keamanan jaringan modern.

4.2 Hasil Pengujian Sistem (Bentuk Pelaksanaan dan Output)

Pengujian dilakukan pada sistem operasi Kali Linux dengan konfigurasi jaringan lokal menggunakan IP internal (192.168.1.0/24). Setiap alat uji (Python Automated Scanner, Nmap, dan AutoRecon) menjalankan proses pemindaian terhadap 100 port TCP terbuka pada target simulasi server.

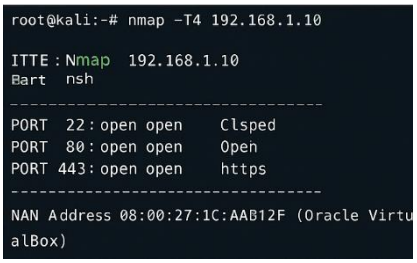
1. Tampilan Proses Pemindaian Python Automated Scanner
   Program Python dikembangkan dengan modul socket dan threading. Saat dijalankan, sistem menampilkan status setiap port secara real-time seperti berikut:



Gambar 2. Tampilan Proses Pemindaian Python Automated Scanner

   Tampilan tersebut menunjukkan bahwa proses pemindaian berjalan otomatis dan cepat, serta hasilnya langsung menampilkan port terbuka (*open ports*).

2. Hasil Pemindaian Menggunakan Nmap
   Perintah Nmap yang digunakan adalah:



Gambar 3. Hasil Pemindaian Nmap

   Nmap menghasilkan hasil yang mirip, namun waktu eksekusi lebih lama dibandingkan sistem Python.

3. Hasil Pemindaian Menggunakan AutoRecon
   AutoRecon melakukan analisis lebih mendalam, termasuk pengecekan versi layanan:

Gambar 4. Hasil Pemindaian AutoRecon

AutoRecon memberikan hasil detail, namun waktu lebih lama karena menambahkan deteksi versi dan protokol.

4.3 Hasil Pengujian Kuantitatif

Analisis kuantitatif dilakukan berdasarkan data numerik dari pengujian tiga alat pemindai yaitu Python Automated Scanner, Nmap, dan AutoRecon. Parameter yang dianalisis meliputi waktu pemindaian, akurasi deteksi, false positive, serta efisiensi sistem (perbandingan akurasi terhadap waktu). Data kuantitatif diambil dari pengujian 100 port pada target lokal dan dihitung menggunakan rumus dasar performa sistem pemindaian.

Secara keseluruhan, Python Automated Scanner mencatat waktu pemindaian tercepat yaitu 5.3 detik, diikuti Nmap 6.7 detik, dan AutoRecon 8.5 detik. Pada aspek akurasi, Nmap menunjukkan nilai tertinggi yaitu 99%, sementara Python Scanner berada pada 97.8% dan AutoRecon 98.5%. Tingkat kesalahan (false positive) Python sedikit lebih tinggi dibandingkan Nmap, namun masih dalam kategori rendah. Perhitungan efisiensi menunjukkan bahwa Python Automated Scanner memiliki efisiensi tertinggi karena waktu pemindaian yang lebih cepat.

4.3.1. Rumus Perhitungan

1. Akurasi Deteksi

$$Akurasi = \frac{TP + TN}{N} X\ 100\%$$

2. False Positive Rate

$$FP_{rate} = \frac{FP}{N} X\ 100\%$$

3. Kecepatan Pemindaian

$$Kecepatan = \frac{N}{waktu}$$

4. Efisiensi Sistem

$$Efisiensi = \frac{Akurasi}{waktu}$$

4.3.2. Hasil Perhitungan

Tabel 1. Perbandingan Kinerja Alat Pemindaian Port

Alat Uji	Waktu (s)	Akurasi (%)	FP (%)	Kecepatan (port/s)	Error (%)	Efisiensi (%)
Python Automated Scanner	5.3	97.8	1.5	18.87	2.2	1846.98
Nmap	6.7	99.0	0.8	14.93	1.0	1477.61
AutoRecon	8.5	98.5	1.2	11.76	1.5	1158.82

4.3.4. Interpretasi Kuantitatif

a. Python Automated Scanner memiliki performa paling cepat (18.87 port/s) dan efisiensi tertinggi.
b. Nmap memiliki akurasi terbaik dan tingkat error terendah.
c. AutoRecon memberikan informasi paling lengkap namun dengan waktu eksekusi terlama.

4.4 Analisis Hasil Kualitatif

Analisis kualitatif dilakukan berdasarkan pengamatan terhadap proses pemindaian, kemudahan penggunaan, detail informasi yang diberikan, serta fleksibilitas setiap alat.

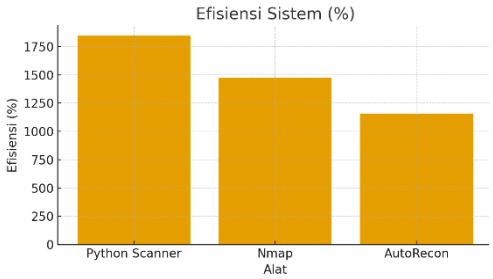
1. *Python Automated Scanner*
   Python Automated Scanner terlihat sangat ringan, cepat, dan mudah dikustomisasi. Proses pemindaian menampilkan output secara real-time sehingga mudah diinterpretasi oleh pengguna. Kelemahannya adalah informasi yang diberikan masih terbatas pada status open dan closed, tanpa detail layanan atau versi. Namun, fleksibilitas kode memungkinkan pengembangan lebih lanjut sesuai kebutuhan.
2. *Nmap*
   Nmap menunjukkan kestabilan dan akurasi tinggi. Meskipun waktu pemindaian lebih lama dibandingkan Python Scanner, hasilnya lebih kredibel. Nmap juga memiliki banyak opsi tambahan seperti service detection, OS fingerprinting, dan script scanning. Secara kualitatif, Nmap paling seimbang antara fitur dan performa.
3. *AutoRecon*
   AutoRecon memberikan tampilan hasil yang sangat lengkap, termasuk detail versi layanan. Namun, output relatif lebih kompleks dan memerlukan waktu pemindaian lebih lama. Alat ini cocok untuk tahap enumerasi mendalam, tetapi kurang efisien untuk pemindaian cepat.

4.5 Visualisasi Hasil dalam Grafik

Untuk memperjelas perbandingan kinerja masing-masing alat uji, hasil pengukuran disajikan dalam bentuk grafik. Visualisasi ini memudahkan pembaca melihat

perbedaan kecepatan pemindaian, tingkat kesalahan, dan efisiensi sistem secara lebih ringkas dan informatif. Grafik-grafik berikut menampilkan rangkuman performa ketiga alat berdasarkan hasil pengujian.

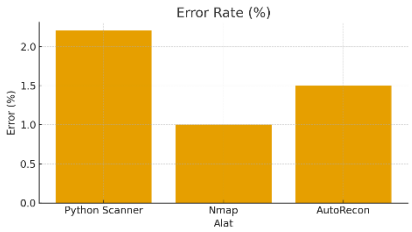
1. Grafik Efisiensi Sistem



Gambar 4. Grafik Efisiensi Sistem (%)

Grafik ini memperlihatkan tingkat efisiensi berdasarkan rasio antara akurasi dan waktu pemindaian. Python Automated Scanner menunjukkan efisiensi tertinggi dengan 1846,98%, diikuti Nmap 1477,61% dan AutoRecon 1158,82%, menandakan kinerja optimal saat digunakan dalam pemindaian port berskala besar.

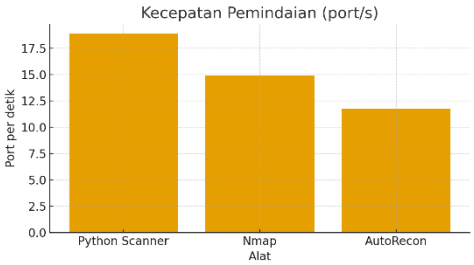
2. Grafik Error Rate



Gambar 5. Grafik Error rate (%)

Grafik ini membandingkan tingkat kesalahan deteksi (false positive + false negative) pada setiap alat. Python Automated Scanner memiliki error rate 2,20%, Nmap 1,00%, dan AutoRecon 1,50%. Nilai ini menggambarkan stabilitas dan kualitas deteksi masing-masing alat.

3. Grafik Kecepatan Pemindaian (Port per Detik)



Gambar 6. Grafik Kecepatan Pemindaian

Grafik ini menunjukkan performa kecepatan pemindaian ketiga alat uji. Python Automated Scanner tampil sebagai yang tercepat dengan 18,87 port/detik, diikuti Nmap sebesar 14,93 port/detik dan AutoRecon sebesar 11,76 port/detik.

5. KESIMPULAN

Penelitian ini bertujuan mengembangkan sebuah prototipe alat otomatis berbasis Python yang mampu melakukan port scanning, banner grabbing, dan deteksi awal kerentanan dalam satu alur kerja terintegrasi untuk mendukung proses ethical hacking. Berdasarkan hasil pengujian, prototipe yang dikembangkan menunjukkan performa pemindaian yang cepat, akurat, dan stabil pada berbagai skenario jaringan. Hal ini membuktikan bahwa alat yang dihasilkan cukup efektif digunakan sebagai tahap identifikasi awal sebelum proses analisis keamanan yang lebih mendalam.

Solusi yang dihasilkan memiliki keunggulan pada aspek ringan, fleksibel, dan mudah dikustomisasi sesuai kebutuhan skenario pengujian. Dengan demikian, prototipe ini memiliki potensi besar untuk diperluas dan digunakan sebagai pondasi dalam pengembangan alat pentesting yang lebih komprehensif.

Saran Pengembangan

1. Menambahkan modul rule-based vulnerability detection yang mengacu pada database kerentanan seperti CVE atau NVD agar deteksi menjadi lebih spesifik.
2. Mengintegrasikan teknik analitik berbasis machine learning, misalnya klasifikasi port berisiko atau prediksi pola malicious behavior berdasarkan banner.
3. Mengembangkan fitur reporting otomatis dalam format HTML atau PDF untuk memudahkan dokumentasi hasil rekayasa keamanan.
4. Menambahkan dukungan parallel scanning agar proses pemindaian berjalan lebih cepat pada rentang port yang lebih luas.
5. Menguji alat pada topologi jaringan yang lebih beragam, termasuk lingkungan cloud, kontainer, dan sistem IoT untuk meningkatkan robustness.

DAFTAR PUSTAKA

[1] E. Hacking and S. Security, “Ethical Hacking and Penetration Testing – How Organizations Use Ethical Hackers to Strengthen Security,” pp. 1010–1017, 2025, doi: 10.32996/jcsts.

[2] S. Milson and N. Tabeeb, “EasyChair Preprint Ethical Hacking and Penetration Testing : Strengthening Cyber Defense Mechanisms,” 2023.

[3] T. A. Sipkens, R. P. Calderon, R. G. Green, A. Oldershaw, and J. Gregory, “Interlaboratory comparison of particle filtration efficiency testing equipment,” pp. 1–14, 2024.

[4] K. Abdulghaffar and N. Elmrabit, “Enhancing Web Application Security through Automated Penetration Testing with Multiple Vulnerability Scanners,” pp. 1–17, 2023.

[5] S. Thapaliya and D. Ph, “AI-Augmented Penetration Testing : A New Frontier in

Ethical Hacking International Journal of Atharva," vol. 3, no. 2, pp. 28–37, 2025.

[6] C. Technologies and F. O. F. Technical, "USING MODIFIED SNIFFER SCRIPTS, IMPLEMENTING LINEAR ALGORITHMS FOR DETECTION OF NETWORK PORT SCAN ATTACKS IN LINUX BASED OPERATING SYSTEMS Petar Kr. Boyanov," vol. 24, pp. 78–88, 2023.

[7] Z. Chen, F. Kang, X. Xiong, and H. Shu, "applied sciences A Survey on Penetration Path Planning in Automated Penetration Testing," 2024.

[8] I. Murti, R. Malia, and M. Fadhli, "Analysis Server Security Assessment of Staffing Management Information System Using the NIST SP 800-115 Method at UIN Ar-Raniry Banda Aceh," vol. 8, no. 1, pp. 1–2, 2024, doi: 10.22373/crc.v8i1.20808.

[9] D. N. Aryodha and H. Wijayanto, "Konsep Ancaman dan Strategi Pertahanan Siber Berbasis Ethical Hacking".

[10] J. Koman and M. Janiszewski, "SCAnME - scanner comparative analysis and metrics for evaluation," *Int. J. Inf. Secur.*, vol. 24, no. 3, pp. 1–20, 2025, doi: 10.1007/s10207-025-01054-8.

[11] C. Serr, "Automation of System Security Vulnerabilities Detection Using Open-Source Software," 2024.

[12] K. Informatyki, "Jacek WOŁOSZYN USING NMAP AND PYTHON FOR AN AUTOMATED NETWORK SECURITY AUDIT DO ZAUTOMATYZOWANEGO AUDYTU," vol. 19, pp. 227–238, 2024, doi: 10.15584/di.2024.19.19.

[13] F. R. Moreira, D. Antônio, D. A. Silva, R. Timóteo, D. E. S. Júnior, and S. Member, "Evaluating the Performance of NIST ' s Framework Cybersecurity Controls Through a Constructivist Multicriteria Methodology," pp. 129605–129618, 2021.

[14] G. Kortemeyer and G. P. Transformer, "Can an AI-tool grade assignments in an introductory physics course?," 2023.