A Framework for Advanced Threat Reconnaissance and Vulnerability Detection

Muhammad Asif[1]
Department of Information and Communication Engineering
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
asifali01623@gmail.com

Talha Shabbir[2]
Department of Information and Communication Engineering
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
talha7504540@gmail.com

Muhammad Basit Rafiq[3]
Department of Information and Communication Engineering
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
muhammadbasitrafiq786@gmail.com

Sana Tariq[4]
Department of Information and Communication Engineering
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
sana.tariq@iub.edu.pk

***Abstract*—This paper compiles a comprehensive review and assessment of a Ruby-based, command-line-based cybersecurity framework that automates the reconnaissance and detection of vulnerabilities (end-to-end) of web and network-based environments. The framework combines the subdomain enumeration, DNS and service discovery, web vulnerability scanning, CMS-specific exploitation checks, content discovery, and reporting into one workflow that is meant to be used by red teams and penetration testers. The primary value of this work lies not in a new detection algorithm, but in a Generation 3 orchestration architecture, which integrates common open-source tools, eliminates glue code, and also standardizes data flow and data output. The system architecture, threat model, and performance characteristics are analyzed in the paper and compared and contrasted with commercial scanners. Case studies on experimental basis, a deliberately vulnerable application and a collection of 100 mixed websites depict that, given the above circumstances, the structure is capable of performing comprehensive reconnaissance and scanning within a matter of a few minutes without causing significant resource usage. Limitation, ethical considerations and future research direction are the last parts of the paper due to the need to integrate authenticated scanning and threat intelligence feeds.**

***Index Terms*—Threat reconnaissance, vulnerability detection, subdomain enumeration, Web vulnerability scanning, penetration testing, offensive security automation, Ruby, Kali Linux.**

I. INTRODUCTION

Cloud-native architecture, applications based on APIs, and distributed web services are changing the landscape of attack surfaces of modern organizations. This has led to reconnaissance and vulnerability assessment being regarded as a part of offensive and defensive programs in cybersecurity [1]-[3]. Some of the industry models like MITRE ATT&CK highlight reconnaissance methods (e.g., active scanning, service discovery, and technology fingerprinting) are the preconditions of numerous later attack methods. As a matter of fact the toolchains used by penetration testers and red teams consist of a series of utilities: subdomain discov- ery (e.g., Amass, Sublist3r), port and service scanning (e.g., Nmap, Masscan), vulnerability scanning (e.g., Nuclei, Wapiti). and CMS-specific tools (e.g. WPScan, CMSMap) [4]-[6]. These tools work alone but they are not in-built. The analysts frequently need to hand-feed data between them, maintain custom scripts and normalize disparate results. This adds to work, leaves room to make mistakes and makes it harder to be reproducible. The framework that is examined in the present paper is going to deal with this gap of integration. It provides: leftmargin=*

- A single command-line tool to coordinate the various open-source tools into a single workflow to reconnaissance and vulnerability detection.
- A hierarchy of architecture that separates input parsing, re- connaissance, scanning, discovery and reporting.
- An interoperable (JSON/HTML) format of reporting that can be processed and integrated further.

This paper will attempt to fulfill two aims: (i) to provide the architecture of this framework and critically evaluate it along with how it fits into the ecosystem of reconnaissance and scanning tools and (ii) to present empirical findings of practical deployments to real-world targets. Orchestration style and workflow design is the novelty and not new detection methods.

II. BACKGROUND AND THREAT LANDSCAPE

The most common assets that constitute the attack surface of most organizations are internet-facing assets such as domains, subdomains, APIs, and object storage. A compromise can be caused by misconfigurations like forgotten subdomains, open administrative panels or even publicly visible storage buckets [7]-[9]. In this respect, reconnaissance includes: leftmargin=*

- Active information search based on publicly available data and CT records.
- Dynamic scanning of DNS records, ports and services.
- Technology fingerprinting in order to determine frameworks, CMSs and WAFs.

DOI: https://doi.org/10.67535/tsp.000002.012
Published under the ISAISS 2026 Proceedings

Existing tooling can be broadly categorized into generations:

A. Generation 1: Reconnaissance-Centric Tools

The tools like Amass and Sublist3r are used to do large-scale pas- sive and active subdomain enumeration [10], [11]. Nevertheless, they usually end with the creation of a list of domains and do not put it in automated port scanning and vulnerability assessment. The operator is left to perform the work of integrating with the need to export the results and invoke other tools and reconcile the output.

B. Generation 2: Scanner-Centric Tools

OWASP ZAP, Acunetix and Nessus are tools that offer deep vulnerability scanning and compliance oriented report ing [12]-[17]. They normally presuppose the knowledge of targets (hostnames or URLs). Others have limited discovery features, and are seldom as extensive as dedicated reconnaissance applications.

C. Generation 3: Orchestration Frameworks

Generation 3 is an attempt to bring together the breadth of Generation 1 reconnaissance and depth of Generation 2 vulnerability scanning, and to offer automation, errorhandling, and standardized reporting [18]-[20]. The design that will be examined here belongs to this category: it coordinates a number of tools that are already present, but does not create the new scanning algorithms. Abstract comparison is given in table I.

TABLE I
CONCEPTUAL COMPARISON OF TOOL GENERATIONS

Generation	Focus	Coverage	Integration Weakness
Gen 1	Reconnaissance	Asset dis- covery	Low No scanning
Gen 2	Scanning	Known targets	Medium Limited dis- covery
Gen 3 (Proposed)	Orchestration	Full work- flow	High External de- pendency

III. FRAMEWORK ARCHITECTURE

It is a layered framework architecture, which includes: (i) Input Layer, (ii) Reconnaissance Engine, (iii) Scanning and Exploitation Modules, (iv) Content and Endpoint Discovery, and (v) Output and Reporting.

A. Input Layer

The command-line parameters that we take as inputs to the input layer are target domain, thread count, wordlists, and the choice of modules and proxy settings. Validation checks make sure that the target syntax is valid and bound. yaml optional configuration files can be used to support workflow and project profiles reuse [4]-[6].

B. Reconnaissance Engine

The reconnaissance engine orchestrates: leftmargin=*

- Passive enumeration: Queries public APIs, CT logs, and DNS datasets [1], [2], [21].
- Active enumeration: Performs wordlist-based bruteforce of subdomains, with wordlists derived from SecLists and context-specific tokens.
- DNS resolution: A pool of DNS resolver threads resolves hostnames and applies wildcard filtering to remove false positives [22]–[24].

• Service pre-fingerprinting: Basic HTTP and TLS probes collect headers and certificate data for later modules [25]–[27]. Results are stored in a normalized structure, including hostname, resolved IPs, inferred technologies, and tags indicating which modules should process the host.

C. Scanning and Exploitation Modules

The scanning layer integrates: leftmargin=*

- Port and service scanning: Port and service scanning Tests of masscan use standard ports, and Nmap NSE scripts detect services and version data [28]-[30].
- Signature-based vulnerability detection: Signaturebased vulnerability detection: Nuclei uses a significant amount of templates of known CVEs over HTTP/TCP/UDP [12], [13], [19].
- Behavioral web scanning: Behavioral web scanning: Wapiti analyses com- mon web application vulnerabilities that are based on OWASP top 10 vulnerabilities [31], [32].
- CMS-specific checks: CMS-based checks: CMSMap and WPScan are activated in the case of technology fingerprints, which identify WordPress, Joomla, or other platforms of this type [33]-[35].

D. Content and Endpoint Discovery

Content discovery contains URL and parameter listing (Arjun), brute-forcing the directory, and finding the exposed configuration or backup files. Other modules in the effort of identifying S3 buckets, directories in repositories (.git) and leaked secrets [7]-[9], [18].

E. Output and Reporting

The framework consolidates all findings into JSON and HTML reports containing: leftmargin=*

- Host and endpoint inventory.
- Identified weaknesses with Nuclei/Wapiti reference.
- CVSS v3.1 vulnerability.
- Aquatone generated screen shots to provide a visual context.

This design enables downstream processing, such as importing results into SIEM or ticketing systems [36]–[38].

IV. ACTIVE RECONNAISSANCE AND BRUTE-FORCE OPTIMIZATION

Active reconnaissance is optimized along three axes.

A. Context-Aware Wordlists

The scheme begins with extensive wordlists (e.g. provided by SecLists) but pruned on TLD, detected nam- ing patterns and strings found in HTML/JavaScript (e.g. names of applications, API prefixes) [39]-[41]. This minimizes the number of unproductive queries besides enhancing the likelihood of finding useful subdomains and directories.

B. Parallel DNS Operations

An up to 50 parallel DNS resolvers are utilized. With a controlled test this setup reached 104 queries per second without strong congestion of upstream infrastruc- ture. Cross-checks are made between A and AAAA records and CNAME records and discarded patterns thought to be indicative of non-unique resolution [25]-[27].

C. Integrated Port Scanning

Masscan is a tool to quickly scan a set of defined common TCP ports. Hosts that respond are then sent to Nmap to be serviced and service version fingerprinted in a more detailed manner that allows the vulnerability scan to be done with specific vulnerability and specific exploits to be chosen to be used [28]–[30]. All of these optimizations improve coverage with the operator-controlling concurrency and rate lim- it [42].

V. VULNERABILITY DETECTION TECHNIQUES

The framework integrates signature based, anomaly based and fuzzing based vulnerability detection.

A. Signature-Based Detection

Nuclei YAML templates offer very accurate signatures of many CVEs (e.g., Log4Shell, Spring4Shell, Heartbleed) [12], [13], [19]. The templates are invoked using identified endpoints using proper protocols and positive matches are known with evidence.

B. Anomaly-Based Checks

Rule-based modules inspect HTTP responses for security misconfigurations such as: leftmargin=*

- Missing HSTS or CSP headers.
- Avoids restrictions on cross-origin scripting; (e.g.,AccessControl-Allow-Origin: *).
- Listing sensitive paths in directories.

Such checks are conditioned by the previous researches of misconfiguration risk in web settings [45]-[47].

C. Fuzzing and Behavioral Analysis

Wapiti is also used to inject vectors of input with curated payloads that attack XSS, SQL injection, command injection, and SSRF [31], [32]. Arjun-based parameter discovery and directory fuzzing extend the coverage even more [39]–[41]. In the framework, where possible, the response-diff based validation is performed in order to minimize the false-positive rate.

TABLE II
QUALITATIVE COMPARISON OF DETECTION TECHNIQUES

Technique	Precision	Recall	Speed
Signature-based	High	Medium	Fast
Anomaly-based	Medium	High	Medium
Fuzzing-based	Medium	Very High	Slow

VI. WEB EXPLOITATION MODULES

In addition to detection, the framework incorporates exploitation modules that are limited in nature and they are used to further validate and examine.

A. CMS Exploitation

Upon detection of CMS instances by CMSMap or WPScan (e.g. WordPress), the framework conducts CMS-specific scans on weak credentials, outdated plugins and known configuration access-related issues [28]-[30], [33]-[35]. It focuses on safe checking as opposed to destructive exploitation.

B. Generic Web Exploits

Default-credential scanners (e.g. Changeme) attempt to log in using a pre-defined set of likely username/password combinations to login inter- faces. Nuclei exploit templates are also a framework capable of invoking explicit proof-ofconcept that requires explicit proof-of-concept and is not prohibited by the rules of engagement [1], [2], [21].

C. Content Discovery and Developer Artefacts

Crawling and directory fuzzing Hakrawler crawling and directory fuzzing find hidden URLs, administrator panels and API endpoints. The additional checks are trying to identify: leftmargin=*

- Unsecured.git repositories and configuration files.
- Backup files (e.g. .bak, .old, compressed archives).
- naming patterns Inferring publicly available S3 buckets.

Tools like TruffleHog are used to scan public repositories to detect leaked secrets, and the results are matched against the target domain [18]-[20]. Key endpoints are created into screenshots in order to present visual evidence towards reporting [4]-[6].

VII. STEALTH CONSIDERATIONS AND EVIDENCE HANDLING

Even though the framework is aimed at authorized testing, there are a number of measures, which minimize the footprint of the operational framework and handle evidence in a safe manner.

A. Traffic Shaping

The framework promotes adjustable request rates and random request delays. The user-agent rotation and concurrency curbs are aimed at preventing the overloading of targets and help to more closely approximate the actual traffic patterns [48]-[50].

B. Evidence Collection

Evidence (e.g., HTTP responses, screenshots, and logs) is collected selectively to avoid excessive storage growth. Traffic captures relevant to detected vulnerabilities are stored alongside the structured findings, enabling later verification without re-running the entire scan [51]–[53].

TABLE III
COMPARISON WITH COMMERCIAL SCANNERS

Feature	Proposed	Burp Suite	Acunetix	Nessus
Reconnaissance	Full	Manual	Limited	Basic
Speed (avg.)	2–3 min	10+ min	5–10 min	15+ min
Cost	Free	Paid	Enterprise	Per scanner
Focus	Red team	Web apps	Compliance	Networks

C. Report Generation and Storage

The reports are produced in small and plain JSON and HTML formats. GPG can be used to apply optional encryption, either before trans- mission or archival in accordance with organizational policies [54]-[56]. The temporary working files are deleted once the scan is complete to minimize left over artefacts [7]-[9].

VIII. PERFORMANCE EVALUATION AND COMPARATIVE ANALYSIS

A. Methodology

Experiments were conducted on: leftmargin=*

- Target A: szic.pk (educational domain).
- Target B: DVWA, a deliberately vulnerable web application.
- Target C: A corpus of 100 mixed domains (.pk and .com) selected for diversity.

On each target, we were able to measure overall execution time, subdomains and endpoints discovered, the number of vulnerability reports (by severity), and the amount of resource used (CPU and RAM). To provide comparisons, we also performed a base workflow using manually chained tools (e.g., Amass + Nmap Nuclei) in like circumstances [39], [41].

B. Results

The input found in szic.pk 12 subdomains and 4 open ports and reported three high-severity vulnerabilities associated with access control and server setup. The overall time was around 152 seconds and the number of worker threads was 50. In the case of DVWA, the framework found 12 out of 15 deliberately introduced vulnerabilities, as compared to 9 vulnerabilities found by a Nuclei- only baseline in the same environment. The 100-domain corpus experiment took about 45 minutes to finish, and the memory consumption was less than 1.5 GB and CPU consumption was less than 80%. The framework reduced the total operator interaction time and enhanced coverage as compared to a scripted baseline in which tools were called upon sequentially, but additional large-scale statistical analysis is needed.

C. Tool Comparison

Table III summarizes a qualitative comparison with common commercial tools.

IX. ETHICAL AND LEGAL CONSIDERATIONS

leftmargin=*

- The framework will be employed in sanctioned security tests. It implements strict target specification through command-line options and also permits the recording of audit actions. Ethical use requires:
- It must have prior written approval of the asset owners.
- Consideration of scope constraints (e.g. public facing assets only).
- Denial-of-service payloads or destructive exploitations are avoided.

Such practices correspond to the NIST guidance on the security testing and evaluation [17] and typical penetration testing codes of conduct.

X. CASE STUDIES AND LIMITATIONS

A. Case Study: Educational Domain (szic.pk)

A legitimate scan of szic.pk suggested several sub domains, an unsecured API endpoint, and poorly configured administration interfaces. A SSRF prone endpoint was also identified. The administrators of the site confirmed the issues and fixed them within 48 hours.

B. Case Study: DVWA

In DVWA, the framework had shown that it could coordinate reconnaissance, parameter discovery and web vulnerability scanning. Additional parameter fuzzing and behavioral analysis explain the greater rate of detection as compared to a Nuclei-only baseline.

C. Limitations

Current limitations include: leftmargin=*

- Particularly, unauthenticated, public-facing is supported; perplexing, authenticated workflows are still not supported.
- Reliance on third-party applications and APIs can be modified or limited in rate.
- Performance claims are informal and founded on few targets; larger-scale benchmarking is beyond work.

XI. CONCLUSION

In this paper, a frame-work based on Ruby and automating several steps in threat reconnaissance and vulnerability detection have been reviewed and assessed. The framework organizes well-defined tools into a consistent workflow, which offers effective gains in terms of operator effort, integration, and reporting. Although the orchestration is mainly the foundation of technical novelty, rather than new detection algorithms, the strategy shows how open-source pieces can be put together to create a Generation 3 offensive security platform. Future efforts will be on the expansion of the empirical assessment, inclusion of authenti- cated scanning and threatintelligence

integration, and creation of graphical interfaces to make the framework more usable by less experienced practitioners.

REFERENCES

[1] F. Abu-Dabaseh and E. Alshammari, "Automated penetration testing: An overview," in *The 4th International Conference on Natural Language Computing*, Copenhagen, Denmark, 2018, pp. 121–129.

[2] M. Mirjalili, A. Nowroozi, and M. Alidoosti, "A survey on web penetration test," *Advances in Computer Science: an International Journal*, vol. 3, no. 6, pp. 107–121, 2014.

[3] A. Alamri, H. Albahri, and R. Ramadan, "Survey on web application security testing methods," *PLOMS AI*, vol. 5, no. 1, 2025.

[4] J. Muniz and A. Lakhani, *Web penetration testing with Kali Linux*. Packt Publishing, 2013.

[5] R. Messier, *Learning Kali Linux: Security testing, penetration testing & ethical hacking*. O'Reilly Media, Inc., 2024.

[6] R. Gowda, "Performance evaluation of automated web vulnerability scanners for cross platforms-red teaming," Ph.D. dissertation, National College of Ireland, Dublin, 2024.

[7] H. Ghazizadeh, G. Tamm, and R. Creutzburg, "Automated tools for cloud security testing," *Electronic Imaging*, vol. 36, pp. 1–7, 2024.

[8] S. Kadam, B. Mahajan, M. Patanwala, P. Sanas, and S. Vidyarthi, "Automated wi-fi penetration testing," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. IEEE, 2016, pp. 1092–1096.

[9] N. Ålander, "Automating vulnerability assessment tools and procedures: Security audit solution for small organizations," 2021.

[10] A. Aruvasagachithan, D. Jeevanandh, T. Bharath, A. Nivas, and R. Mohana Santhiya, "Automating web application penetration testing using ai."

[11] S. Putit and L. Y. Bujang Khedif, "Exploring vega: a tool for scanning vulnerabilities in penetration testing within web applications," *Borneo Akademika*, vol. 8, no. 2, pp. 176–187, 2024.

[12] I. V. Stetsenko and V. Savchuk, "Information system penetration testing using web attack automated simulation," in *International Conference on Computer Science, Engineering and Education Applications*. Springer, 2020, pp. 396–406.

[13] E. Andersson, "Automated virtualization in digital forensic and penetration testing work," 2019.

[14] S. Camtepe, K. Bsufka, L. Hennig, C. Simsek, and S. Albayrak, "A framework for automated identification of attack scenarios on it infrastructures," *PIK: Praxis der Informationsverarbeitung und Kommunikation*, vol. 35, no. 1, pp. 25–31, 2012.

[15] A. Astély and J. Ekroth, "Penetration testing ten popular swedish android applications," 2022.

[16] A. Efe, "A risk assessment on usage of kali tools to hack and manipulate web-based mis and erp applications," *Yönetim Bilişim Sistemleri Dergisi*, vol. 11, no. 1, pp. 62–80.

[17] K. Scarfone, *Technical guide to information security testing and assessment: recommendations of the National Institute of Standards and Technology*. DIANE Publishing, 2009.

[18] Y. Tyagi, S. Bhardwaj, S. Shekhar *et al.*, "Efficient vulnerability assessment and penetration testing: A framework for automation," in *2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)*. IEEE, 2023, pp. 553–557.

[19] M. P. Shah, "Comparative analysis of the automated penetration testing tools," Ph.D. dissertation, National College of Ireland, Dublin, 2020.

[20] H. V. Solanki, "Limiting attack surface for infrastructure applications using custom yaml templates in nuclei automation," Ph.D. dissertation, National College of Ireland, Dublin, 2023.

[21] P. Modesti, L. Golightly, L. Holmes, C. Opara, and M. Moscini, "Bridging the gap: A survey and classification of research-informed ethical hacking tools," *Journal of Cybersecurity and Privacy*, vol. 4, no. 3, pp. 410–448, 2024.

[22] P. Segec, R. Kaloc, and R. Dobis, "Portable system for automated audit of wi-fi networks in organizations," in *2021 19th International Conference on Emerging eLearning Technologies and Applications (ICETA)*. IEEE, 2021, pp. 333–338.

[23] M. Kurek and M. Purwin, "Automatic testing of web services based on wsdl document," 2014.

[24] J. A. Plot, "Red team in a box (RTIB): Developing automated tools to identify, assess, and expose cybersecurity vulnerabilities in Department of the Navy systems," 2019.

[25] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th International Conference on Software Engineering*, 2018, pp. 303–314.

[26] H. Jiang, T. Choi, and R. K. Ko, "Pandora: A cyber range environment for the safe testing and deployment of autonomous cyber attack tools," in *International Symposium on Security in Computing and Communication*. Springer, 2020, pp. 1–20.

[27] G. Sharma, S. Vidalis, C. Menon, and N. Anand, "Analysis and implementation of semi-automatic model for vulnerability exploitations of threat agents in NIST databases," *Multimedia Tools and Applications*, vol. 82, no. 11, pp. 16951–16971, 2023.

[28] J. Grunwald, "Stronghold: Automating corporate security," 2023.

[29] S. Sankarapandian, "Detecting exploitable vulnerabilities in android applications," Ph.D. dissertation, University of Waterloo, 2021.

[30] W. Holt, R. Dawson, and H. Agoro, "Development of an automated digital forensics toolkit for incident response," 2021.

[31] G. B. Gaggero, A. Fausto, F. Patrone, and M. Marchese, "A framework for network security verification of automated vehicles in the agricultural domain," in *2022 26th International Conference Electronics*. IEEE, 2022, pp. 1–5.

[32] S. Bodkhe, M. Jadhav, and S. Warkar, "Metasploit for exploit automation and threat detection on linux."

[33] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," *ACM Computing Surveys (CSUR)*, vol. 44, no. 2, pp. 1–42, 2008.

[34] A. M. Algamdi, *An assessment of Cloud models against security vulnerabilities using a semi-automated vulnerability test model*. University of Houston-Clear Lake, 2014.

[35] J. Kinyua and L. Awuah, "Ai/ml in security orchestration, automation and response: Future research directions," *Intelligent Automation & Soft Computing*, vol. 28, no. 2, 2021.

[36] C. Mainka, J. Somorovsky, and J. Schwenk, "Penetration testing tool for web services security," in *2012 IEEE Eighth World Congress on Services*. IEEE, 2012, pp. 163–170.

[37] A. S. Moselekatsi and S. Doss, "A study on ethical hacking and penetration testing."

[38] O. B. Fredj, O. Cheikhrouhou, M. Krichen, H. Hamam, and A. Derhab, "An owasp top ten driven survey on web application protection methods," in *International Conference on Risks and Security of Internet and Systems*. Springer, 2020, pp. 235–252.

[39] R. Mahmood, J. Pennington, D. Tsang, T. Tran, and A. Bogle, "A framework for automated api fuzzing at enterprise scale," in *2022 IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE Computer Society, 2022, pp. 377–388.

[40] D. E. Simos, "The mathematics behind automated penetration testing framework," in *SBA Research Security Forum*, 2014.

[41] M. Andreolini, A. Artioli, L. Ferretti, M. Marchetti, M. Colajanni, C. Righi *et al.*, "A framework for automating security assessments with deductive reasoning," in *CEUR Workshop Proceedings*, vol. 3488. CEUR-WS, 2023.

[42] A. Doraiswamy, *Security testing handbook for banking applications*. IT Governance Ltd, 2009.

[43] C. Cowan, C. Pu, D. Maier, J. Walpole, P. Bakke, S. Beattie, A. Grier, P. Wagle, Q. Zhang, and H. Hinton, "Stackguard: Automatic adaptive detection and prevention of buffer-overflow attacks," in *USENIX Security Symposium*, vol. 98, San Antonio, TX, 1998, pp. 63–78.

[44] S. Narayanan and S. A. McIlraith, "Simulation, verification and automated composition of web services," in *Proceedings of the 11th international conference on World Wide Web*, 2002, pp. 77–88.

[45] C. Kumar, "Net-tok: Network security tool-kit for network administrators."

[46] J. Gregory and Q. Liao, "Autonomous cyberattack with security-augmented generative artificial intelligence," in *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*. IEEE, 2024, pp. 270–275.

[47] L. Wu, X. Zhong, J. Liu, and X. Wang, "Ptgroup: An automated penetration testing framework using llms and multiple prompt chains," in *International Conference on Intelligent Computing*. Springer, 2024, pp. 220–232.

[48] S. Nandi, "Evaluating the effectiveness of security testing tools in automated testing," 2024.

[49] M. Albahar, D. Alansari, and A. Jurcut, "An empirical comparison of pen-testing tools for detecting web app vulnerabilities," *Electronics*, vol. 11, no. 19, p. 2991, 2022.
[50] Y. Alkhurayyif and Y. S. Almarshdy, "Adopting automated penetration testing tools: A cost-effective approach to enhancing cybersecurity in small organizations," *Journal of Information Security and Cybercrimes Research*, vol. 7, no. 1, pp. 51–66, 2024.
[51] E. Filiol, F. Mercaldo, and A. Santone, "A method for automatic penetration testing and mitigation: A red hat approach," *Procedia Computer Science*, vol. 192, pp. 2039–2046, 2021.
[52] M. C. Ghanem, "Towards an efficient automation of network penetration testing using model-based reinforcement learning," Ph.D. dissertation, City, University of London, 2023.
[53] K. C. Goh, "Toward automated penetration testing intelligently with reinforcement learning," Ph.D. dissertation, National College of Ireland, Dublin, 2021.
[54] A. Roshini, N. Suganthi, K. Karthika *et al.*, "An efficient penetration testing framework for web applications," in *2025 3rd International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA)*. IEEE, 2025, pp. 1–14.
[55] P. S. Nikam, "Customized penetration testing framework for assessing the security of amazon web services (aws)," Ph.D. dissertation, National College of Ireland, Dublin, 2024.
[56] F. Viggiani, "Design and implementation of a non-aggressive automated penetration testing tool: An approach to automated penetration testing focusing on stability and integrity for usage in production environments," Master's thesis, Institutt for telematikk, 2013.
[57] E. R. Narwal and G. Gupta, "Tracks covering in penetration testing and cyber attack."