ISSN: 2664-9640

A Survey on the Evolution of Automated Penetration Testing Techniques

Chen Wang, Yuxia Fan, Jingjing Wu, and Yueqin Li*

School of Beijing Union University, Beijing, China

*Corresponding Author: Yueqin Li

Abstract

As networked systems grow in scale and complexity, traditional penetration testing remains constrained by its reliance on human expertise, substantial labor costs, and limited suitability for continuous assessment. These limitations have driven penetration testing toward greater automation and autonomy.This review examines representative research on automated penetration testing published between 2002 and 2025. It traces the evolution of the field across four major paradigms: formal attack knowledge representation and reasoning, automated vulnerability analysis and attack execution, reinforcement learning-based autonomous decision-making, and large language model-based generative agents.The literature shows a clear shift from predefined, rule-based attack-path generation toward adaptive policy learning and general-purpose reasoning in dynamic environments. Nevertheless, significant challenges remain in real-world generalization, long-horizon planning, persistent knowledge and state management, standardized evaluation, and safe operation.Finally, we discuss emerging directions, including knowledge-enhanced architectures, the integration of reinforcement learning with large language models, human–agent collaboration, and autonomous penetration testing in realistic network environments.

Keywords

Automated Penetration Testing; Attack Graph; Reinforcement Learning; Large Language Model; Agent.

1. Introduction

The ongoing development of cloud computing, the Internet of Things and complex network infrastructure have increased the attack surface of current systems and provided more avenues for attack. Penetration testing will be carried out to determine how vulnerable the system is in a real-world scenario by simulating an attack. The steps are generally planning, reconnaissance, exploitation and reporting. Vulnerability scanning is now highly automated; however, human experts are still needed to analyse the results and determine how serious they are[1]. The current automated penetration-testing methods are also highly scenario-dependent. Their accuracy, adaptability and support for end-to-end automation are still relatively weak[2]. Therefore, the new problem is no longer about automated tool operation. Now the field studies systems that can reach a goal state, choose an appropriate path, and adapt behaviour under conditions of uncertainty and change.

Recently, most researchers have moved away from explicit knowledge representation and begun exploring autonomous decision-making. The first type of attack graph shows a target with certain weaknesses and prerequisites, but not network conditions. The above representations enabled machine-processable reasoning over multi-stage attack paths and incorporated uncertainty into the planning process[3-5]. Program analysis and automated vulnerability discovery have reached a high degree of automation, and now include operational

ISSN: 2664-9640

attack execution. Fuzzing, symbolic execution and autonomous cyber reasoning systems helped build closed-loop connections between vulnerability discovery, exploitation and environmental feedback[6-7]. At the same time, MITRE ATT&CK provided an organised structure for knowledge on adversary tactics and techniques. The above frameworks provided a common semantic basis for attack modelling, red teaming and knowledge organisation of autonomous systems[8]. The above development laid the foundation for machine-driven attack reasoning but was still relatively dependent on pre-defined knowledge and environmental models.

Reinforcement learning is a different decision mode that can make better choices by having an agent interact repeatedly with an environment. This model is naturally suitable for penetration testing, and each observation and action will impact the following opportunities. Therefore, RL and DRL have been applied to attack path selection, policy reuse, post-exploitation decision-making and large-scale network optimisation[9-13]. Simulation Platforms and Cyber-agent Training Environments have also helped to accelerate policy learning. Recently, more attention has been paid to whether the learned good policies for a fixed environment can be generalised to unseen targets and changing situations. Therefore, the main problems in RL-based penetration testing now are generalisation and simulation-to-real transfer[14].

With the appearance of large language models, research in this field has moved from policy optimisation in a fixed-state-space to general-purpose semantic reasoning and tool orchestration. LLMs can understand natural language goals, process scanner and tool outputs, use security knowledge, and output the next steps. As a result, the function of the human tester has been expanded to support the generation of tasks and tool applications by a generative agent[15-18]. Increased Flexibility comes with a set of new restrictions. LLM-based agents may lose the original context after many operations, return invalid actions, or misapply security tools. They are also unreliable for planning complex multi-stage attack chains. At present, end-to-end evaluations do not show sufficient stability for real-world application of full autonomy.

Thus, the development of automated penetration testing should not be regarded as changing one algorithm to another. Instead, the field has gradually altered how attack knowledge, environmental conditions, decision-making and execution are shown and linked. Attacks graphs, automated planning, reinforcement learning and generative agents have all shown remarkable progress. However, the literature has not provided a single definition for the scope and autonomy of "automated penetration testing". Many methods are also only tested in simulators or controlled testbeds, and there is a gap between attack planning and real-system operation[19]. Therefore, this paper will take the development of decision-making mechanisms and autonomous capabilities as its reference system for analysis. Representative studies published from 2002 to 2025 are selected, and the transition from explicit attack knowledge and formal planning to learned policies and LLM-based agents is traced. We will compare the above paradigms in terms of knowledge representation, decision-making mechanism, automation scope, generalisation and real-world application. Finally, recent end-to-end benchmarks are introduced to identify the main deficiencies in the current autonomous penetration testing and to suggest future research directions.

2. Evolution of Automated Penetration Testing Technologies

Automated penetration testing relies on two closely coupled capabilities. The first is attack planning, which determines suitable targets and action sequences from available environmental information. The second is attack execution, which converts those plans into concrete operations through scanning, exploitation, and post-exploitation tools. As artificial intelligence has advanced, these capabilities have evolved from relatively independent automation modules toward closed-loop systems that perceive, plan, execute, and adapt.

ISSN: 2664-9640

2.1. Attack Graphs and Formal Reasoning

Early work focused on presenting network vulnerabilities, system states and adversary actions in forms suitable for automated reasoning. Sheyner and others have introduced an automatic way to build and analyze attack graphs[3]. Their method represents the transition of attack actions and system states as a graph and finds feasible paths to the target state. Attack graphs have thus extended the scope of security analysis to address multiple vulnerabilities and attack sequences simultaneously.

Ou and others have extended this research direction to MulVAL, and they use Datalog to represent vulnerabilities, host configurations, network connections and privilege relationships in a unified model[4]. Logical Rules are used to derive dependencies among security conditions. MulVAL collects vulnerability data, scan results and network information to present an all-in-one reasoning system for the identified deficiencies rather than showing them separately. Support multiple hosts and multi-stage attack analysis for the whole network.

Attack graphs and logical reasoning generally assume that some information about the environment is available. However, real penetration tests are often carried out with incomplete knowledge. Sarraute and others addressed this problem by modelling penetration testing as a partially observable Markov Decision Process[5]. Their formation includes both observations obtained by scanning and actions related to exploitation. Therefore, the agent will decide whether to obtain more information or carry out an attack based on its current belief state. POMDPs are more natural representations of uncertainty, but exact planning becomes prohibitively expensive with larger networks. Network decomposition and related approximations are thus required to maintain tractability.

At that time, the foundation for automated penetration testing was being laid. Systems can derive attack sequences from network conditions, vulnerability status and other reasons with relatively high interpretability. However, their effectiveness directly related to the richness of the pre-defined vulnerability knowledge and state descriptions. Therefore, they were not very suitable for an open and changing environment.

2.2. Automated Vulnerability Analysis and Autonomous Attacks

Attack Planning selects the target and the order of attack, and advanced automation includes vulnerability discovery and exploitation. Stephens et al. proposed Driller, which combines fuzzing with selective concolic execution[6]. Fuzzing cannot find a new path because of the complex constraints, so symbolic execution is used to solve the corresponding conditions. Then the new input is given to the fuzzer. The two methods have different strengths and are used together. Reduce the cost of unrestricted symbolic execution and improve access to deeper program paths.

DARPA's Cyber Grand Challenge has sped up the development of vulnerability-analysis systems for autonomous cyber reasoning. Mechanical Phish integrated program analysis, vulnerability discovery, exploitation, patch generation and feedback-driven decision-making in a Cyber Reasoning System[7]. It can continuously identify goals and adjust behaviour independently of people. The above systems are capable of running binary programs in a competition. However, their perception-analysis-execution-feedback loop provided a necessary practical basis for the following autonomous penetration-testing architecture.

A related research direction examines how to depict the knowledge of the adversary. MITRE ATT&CK organises observed adversary behaviour into tactics, techniques and sub-techniques [8]. Vulnerability databases focus on particular software defects, while ATT&CK details the tactics and techniques that attackers use in the course of a cyberattack. This form is now suitable for red teaming, adversary emulation and high-level planning of autonomous security systems.

ISSN: 2664-9640

The above work is now an operational-level attack execution rather than a static analysis. Only a small number of targets or highly controlled environments have been used to test automated vulnerability discovery and autonomous cyber reasoning. Therefore, they did not yet constitute full-fledged enterprise penetration tests and may require reconnaissance, initial access, privilege escalation, credential usage, and lateral movement.

2.3. Reinforcement Learning-Driven Autonomous Decision-Making

Reinforcement learning enables agents to improve their behavior through repeated interaction with an environment and feedback on long-term outcomes. This learning paradigm is well suited to penetration testing because each action changes the information and opportunities available to the tester. Ghanem and Chen modeled network penetration testing as a POMDP and explored the learning and reuse of attack policies[9]. This approach moved beyond static attack-path computation by allowing systems to accumulate and exploit experience from previous testing episodes.

Deep reinforcement learning further enabled complex states and action spaces to be approximated with neural networks. Hu et al. combined attack information generated by MulVAL with a DQN to support attack-path selection[10]. The approach improved policy search in more complex environments. However, the available actions remained constrained by the attack model constructed in advance.

The growth of RL-based research also created a demand for standardized training environments. CybORG provides a common interface across simulated and emulated cyber environments[11]. Agents can therefore perform extensive low-cost training before being evaluated in environments that more closely resemble real systems. This design highlights an important challenge for autonomous cyber agents. Simulators inevitably omit or simplify characteristics of operational systems, producing a persistent simulation-to-reality gap.

RL-based penetration testing also expanded beyond initial access and path selection. Maeda and Mimura integrated deep reinforcement learning with a post-exploitation framework[12]. Their agent selected post-exploitation actions according to the privileges already obtained and the current target state. The decision problem thus expanded from gaining initial access to determining how control should be extended after a compromise.

Scalability became another major concern as network size and action spaces increased. Ghanem et al. applied hierarchical reinforcement learning to decompose large penetration-testing problems into smaller subproblems[13]. This reduced the complexity of the overall POMDP. Such hierarchical approaches suggest that large-scale autonomous testing requires more than stronger learning algorithms. Effective abstraction of network structure and action spaces is also necessary.

More recent work has focused on whether learned policies generalize beyond the environments used during training. Venturi et al. compared DQN, PPO, and A2C on target hosts that were not included in the training set[14]. Their findings indicate that some DRL agents can generalize to new host configurations under controlled conditions. Nevertheless, current evaluations commonly assume that the underlying sets of ports, services, and vulnerability classes are known during training. Generalization to genuinely novel configurations, adaptive defenses, and real enterprise networks remains insufficiently demonstrated.

The major contribution of reinforcement learning is therefore a transition from predefined path search to adaptive closed-loop decision-making. The resulting process combines environmental observation, policy selection, execution feedback, and policy optimization. However, performance remains highly sensitive to state representation, reward design, training environments, and the predefined action space.

ISSN: 2664-9640

2.4. Large Language Model-Driven Generative Agents

Reinforcement Learning generally requires the explicit definition of states, actions and rewards. In actual penetration tests, much of the relevant information is in a natural language form, terminal output, source code, configuration data and unstructured security knowledge. Large language models can perform semantic reasoning, code generation and extensive pre-training on the above types of information.

Happe and Cito were among the first to study LLMs as intelligent assistants for penetration testers[15]. A high-level task plan and a low-level vulnerability analysis of the experiment were available, and feedback could be obtained from the target environment. The scope of this direction also includes automatic command generation. LLMs can also interpret environmental evidence based on security knowledge and propose the next test.

PentestGPT has been extended to a multi-stage penetration testing workflow for LLMs[16]. Separates task reasoning, command generation and output parsing in cooperating modules. This architecture can reduce some of the problems of information management in an expanding interaction history. Based on the above tests, LLMs can use some security tools, interpret the results, and suggest follow-up actions. Complex targets have the same defects as before: loss of early information, sticking to an unproductive path, and wrong tool selection.

Research has since moved away from individual frameworks to a system of capability boundaries. Isozaki et al. introduced an all-encompassing benchmark for LLM-based penetration testing and evaluated the performance of reconnaissance, exploitation and privilege escalation[17]. Persistent summarization and structured task management and retrieval augmentation have also been used to extend the long-term effect. The above development shows that LLM-based penetration-testing agents are no longer context-limited. Newer systems are gradually adding external memory, task-management modules and knowledge-retrieval mechanisms to the language model.

AutoPenBench is also an all-in-one evaluation environment for generative agents[18]. Add an introduction to the system of documented vulnerabilities and use intermediate goals to find out where the agent fails during execution. According to the above results, full-automation and human-assisted modes have different reliability levels. LLMs are thus expanding the scope of knowledge and interaction for automated penetration testing. Long-term planning, stable execution and generalisation in a complex environment have not been achieved.

Skandylas and Asplund addressed the problem in the form of complementary architecture[19]. Their work integrates attack planning, runtime decision-making and tool execution in a unified adaptive loop, and evaluates ADAPT with actual penetration-testing tools and virtual networks. Therefore, the future development of this work may not rely on improving a single attack algorithm. Therefore, a good system should integrate knowledge representation, environmental perception, decision-making, tool execution and persistent feedback in an end-to-end manner.

3. Comparative Analysis and Current Research Status

The preceding evolution shows that the major paradigms differ in more than their underlying algorithms. They also differ in how they represent environments, obtain decision information, and define the scope of autonomous operation. Table 1 summarizes these distinctions.

ISSN: 2664-9640

Table 1. Comparison of Major Automated Penetration Testing Paradigms

Technical paradigm	Representative studies	strengths	limitations
Attack graphs and logical reasoning	Sheyner et al.[3], MulVAL[4]	Strong interpretability and explicit attack dependencies	Dependence on relatively complete environment models and predefined knowledge
Uncertainty-aware attack planning	Sarraute et al.[5]	Explicit treatment of incomplete information	Poor scalability with large state spaces
Automated vulnerability analysis and attack execution	Driller[6], Mechanical Phish[7]	High automation of vulnerability discovery and attack execution	Restricted target classes and controlled evaluation environments
RL/DRL	Ghanem et al.[9], Hu et al.[10], Maeda et al.[12]	Adaptive policy learning from environmental feedback	Strong dependence on state representation, reward design, and training environments
LLM-based agents	Happe et al.[15], PentestGPT[16]	Broad knowledge coverage and flexible interaction with heterogeneous tools	Weaknesses in long-horizon planning, hallucination, and execution reliability
Adaptive integrated architectures	ADAPT[19]	Integration of planning and attack execution in a closed loop	Capability remains limited by available tools and knowledge coverage

3.1. From Automated Execution to Autonomous Decision-Making

The original Security Automation generally addresses repetitive operations that are prone to manual intervention. Attack graphs and automated planning extended the automation of individual actions to sequences of actions and attack paths. Reinforcement Learning can be used to learn attack policies based on environmental feedback. An LLM-based agent will automatically perform task recognition, knowledge retrieval, tool selection and result analysis.

Thus, the entire course can be summarised as follows:

Automated tool execution → Automated attack-path planning → Autonomous attack-policy learning → General-purpose agent reasoning.

The Development will also alter the metrics for judging automated penetration tests. Scanning Speed and the quantity of found vulnerabilities are no longer the sole indicators. Determine the degree of autonomy for data and plan handling, error handling, bad decision correction, etc., based on evaluation.

3.2. From Local Automation to End-to-End Attack Workflows

Early approaches generally automated only one component of the penetration-testing process. Attack graphs focused on reachability and attack paths, whereas automated vulnerability discovery focused on software flaws. Early RL systems mainly addressed attack selection. More recent work has extended automation to post-exploitation, privilege escalation, task planning, and tool orchestration.

Automating individual stages, however, does not imply that the entire penetration test has been automated. A complete system must support both attack planning and attack execution. It must also revise its strategy when new information becomes available at runtime[19]. The literature should therefore distinguish among tool automation, intelligent assistance, semi-autonomous testing, and end-to-end autonomous testing. The use of AI alone is not a sufficient measure of autonomy.

ISSN: 2664-9640

3.3. The Persistent Gap Between Simulation and Real-World Networks

Controlled simulation environments provide RL agents with large amounts of inexpensive interaction data. At the same time, agents may overfit to particular network structures, vulnerability distributions, or reward functions. Operational networks exhibit configuration diversity, changing services, defensive mechanisms, and unknown vulnerabilities that are difficult to reproduce faithfully in simulation.

DRL studies have begun to evaluate generalization by separating training targets from unseen test hosts[14]. However, generalization to unseen test instances is not equivalent to deployment in open real-world environments. A similar limitation affects LLM-based agents. Most current evaluations rely on CTF challenges, vulnerable virtual machines, or publicly documented CVEs. Multi-host, multi-stage enterprise attacks and interactions with dynamic defenses remain underrepresented[18].

High success rates in simulation therefore cannot be interpreted directly as evidence of practical deployability. Future evaluation frameworks must balance three competing requirements: reproducibility, operational safety, and sufficient environmental realism.

4. Major Challenges and Future Research Directions

4.1. Generalization and Continuous Adaptation in Unknown Environments

Operating systems, services, configurations, and vulnerability combinations change continuously in real networks. Attack-graph and RL approaches generally assume that relevant states, actions, or attack knowledge can be specified in advance. When a target contains services, vulnerabilities, or defensive mechanisms absent from training, learned policies may lose effectiveness.

Future research should therefore move beyond optimizing policies for fixed environments and toward continuous adaptation in open settings. A promising direction is to combine the broad knowledge generalization of LLMs with the feedback-driven optimization of reinforcement learning. LLMs could interpret unfamiliar services and unstructured tool outputs, while RL could adjust action selection using runtime feedback from the target environment. Such integration may reduce dependence on fixed state spaces and manually encoded rules.

4.2. Long-Horizon Planning and Persistent State Management

Complete penetration tests consist of many interdependent actions. Information obtained during reconnaissance, including hosts, services, credentials, and vulnerabilities, may become relevant only much later during privilege escalation or lateral movement. LLM-based agents can lose such information when relying entirely on conversational context. This can lead to repeated actions, inconsistent states, or continued exploration of unproductive paths[16-17].

Future systems therefore require persistent external memory and state-management mechanisms. Assets, ports, credentials, vulnerabilities, privileges, historical actions, and execution outcomes can be maintained in structured stores. Attack graphs, task trees, or adversary knowledge frameworks can then represent relationships among these elements. Under this architecture, the LLM can focus on high-level reasoning while external components maintain state consistency and task progress.

4.3. Standardized Benchmarks and Evaluation Frameworks

Existing studies use substantially different network topologies, numbers and types of vulnerabilities, reward functions, and success criteria. These differences make direct comparison difficult. RL studies often report cumulative reward, attack steps, or convergence behavior. Evaluations of LLM-based agents more often emphasize task completion or final attack success.

ISSN: 2664-9640

AutoPenBench and related efforts have begun to address this problem through public tasks, standardized interfaces, and intermediate milestones[18]. Future benchmarks should additionally include multi-host attacks, dynamic environments, unknown vulnerabilities, and defensive feedback. Evaluation should consider task success, coverage, invalid actions, execution time, computational cost, operational risk, and reproducibility. A unified evaluation framework is necessary to determine whether higher benchmark performance actually corresponds to greater autonomous testing capability.

4.4. Knowledge-Enhanced and Hybrid Intelligent Architectures

The major technical paradigms have complementary strengths. Attack graphs and logical rules provide interpretability but require explicit knowledge. Reinforcement learning adapts through feedback but is sensitive to its training environment. LLMs offer broad knowledge transfer and semantic reasoning but remain vulnerable to hallucination and inconsistent execution.

Future systems are therefore likely to adopt hybrid architectures. Knowledge frameworks such as ATT&CK can provide standardized semantics for adversary behavior[8]. Attack graphs can encode preconditions and causal dependencies among attack steps. LLMs can support high-level task interpretation, knowledge retrieval, and plan generation. Reinforcement learning can optimize action selection using environmental feedback. Combining structured constraints with generative reasoning and runtime feedback may support a more reliable closed loop of knowledge, planning, execution, and adaptation.

4.5. Human–Agent Collaboration and Controlled Autonomy

Greater autonomy allows penetration-testing systems to execute potentially disruptive security operations with less direct supervision. Safety boundaries, authorization controls, and traceability must therefore be treated as core architectural requirements. The Menlo Report identifies Respect for Persons, Beneficence, Justice, and Respect for Law and Public Interest as key ethical principles for ICT research[20]. It also emphasizes harm assessment, legal compliance, and accountability.

At the current level of technological maturity, human–agent collaboration is likely to be more practical than fully unattended operation. Agents can perform information gathering, result organization, routine testing, and generation of candidate attack strategies. Human testers can retain authority over scope definition, anomaly assessment, and approval of high-risk actions. Risk-based action classification, human confirmation, least-privilege execution, and comprehensive logging can improve efficiency while reducing the consequences of incorrect or out-of-scope actions.

5. Conclusion

In the past 20 years, automated penetration testing tools have been updated and now operate in a relatively autonomous mode. Attack graphs and logical reasoning formalised vulnerability dependencies and multi-stage attack paths. POMDP-based methods added uncertainty to attack planning. Automated Vulnerability Analysis and Cyber Reasoning Systems have shown closed-loop attack execution. Reinforcement learning then enabled systems to learn and refine attack policies through environmental interaction. Recently, large language models have integrated general security knowledge, semantic reasoning and flexible tool use to promote the development of generative penetration-testing agents.

The most significant change at all these times is not an improvement in the performance of a particular attack algorithm. Fundamentally speaking, the representation of attack knowledge, environmental state and decision-making have also changed. The Area has transitioned from explicit rules to data- and feedback-driven policies, and more recently, to knowledge-rich

ISSN: 2664-9640

generative reasoning. At the same time, the scope of automation has also expanded from vulnerability scanning and path planning to exploitation, post-exploitation and end-to-end task execution.

Several barriers still prevent practical application, such as reliance on training environments, limited generalisation to operational networks, weak long-horizon state management, inconsistent evaluation practices, and the safety risks of autonomous attack execution. The future system will likely integrate the structured knowledge of attack graphs and ATT&CK with LLM reasoning, reinforcement-learning-based policy optimisation, and feedback from the real world. The above structures can support agents with permanent memory, dynamic planning and continuous adaptation. To improve the reliability, interpretability and controllability of standardised benchmarks, human-agent collaboration and explicit safety constraints will also be added. Together, the above developments will enable automated penetration testing to move from research prototypes to practical security assessment tools

Acknowledgments

Not applicable.

References

[1] Scarfone, K., Souppaya, M., Cody, A., & Orebaugh, A. (2008). Technical guide to information security testing and assessment. NIST Special Publication 800-115, 2-25.

[2] Saber, V., Bahaa-Eldin, A. M., ElSayad, D., & Fayed, Z. T. (2023). Automated penetration testing, a systematic review [Conference paper]. 2023 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), 373-380.

[3] Sheyner, O., Haines, J., Jha, S., Lippmann, R., & Wing, J. M. (2002). Automated generation and analysis of attack graphs [Conference paper]. Proceedings 2002 IEEE Symposium on Security and Privacy, 273-284.

[4] Ou, X., Govindavajhala, S., & Appel, A. W. (2005). MulVAL: A logic-based network security analyzer [Conference paper]. Proceedings of the 14th USENIX Security Symposium, 113-128.

[5] Sarraute, C., Buffet, O., & Hoffmann, J. (2012). POMDPs make better hackers: Accounting for uncertainty in penetration testing. Proceedings of the AAAI Conference on Artificial Intelligence, 26(1), 1816-1824.

[6] Stephens, N., Grosen, J., Salls, C., Dutcher, A., Wang, R., Corbetta, J., Shoshitaishvili, Y., Kruegel, C., & Vigna, G. (2016). Driller: Augmenting fuzzing through selective symbolic execution [Conference paper]. Proceedings of Network and Distributed System Security Symposium (NDSS 2016), 1-16.

[7] Shoshitaishvili, Y., Bianchi, A., Borgolte, K., Cama, A., Corbetta, J., Disperati, F., Dutcher, A., Grosen, J., Grosen, P., Machiry, A., & others. (2018). Mechanical phish: Resilient autonomous hacking. IEEE Security & Privacy, 16(2), 12-22.

[8] Strom, B. E., Applebaum, A., Miller, D. P., Nickels, K. C., Pennington, A. G., & Thomas, C. B. (2018). MITRE ATT&CK: Design and philosophy [Technical report]. MITRE Corporation.

[9] Ghanem, M. C., & Chen, T. M. (2020). Reinforcement learning for efficient network penetration testing. Information, 11(1), 6.

[10] Hu, Z., Beuran, R., & Tan, Y. (2020). Automated penetration testing using deep reinforcement learning [Conference paper]. 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), 2-10.

[11] Standen, M., Lucas, M., Bowman, D., Richer, T. J., Kim, J., & Marriott, D. (2021). CybORG: A gym for the development of autonomous cyber agents [Preprint]. arXiv. https://arxiv.org/abs/2108.09118

[12] Maeda, R., & Mimura, M. (2021). Automating post-exploitation with deep reinforcement learning. Computers & Security, 100, 102108.

ISSN: 2664-9640

[13] Ghanem, M. C., Chen, T. M., & Nepomuceno, E. G. (2023). Hierarchical reinforcement learning for efficient and effective automated penetration testing of large networks. Journal of Intelligent Information Systems, 60(2), 281-303.

[14] Venturi, A., Andreolini, M., Marchetti, M., & Colajanni, M. (2024). Assessing generalizability of deep reinforcement learning algorithms for automated vulnerability assessment and penetration testing. Array, 24, 100365.

[15] Happe, A., & Cito, J. (2023). Getting pwn'd by AI: Penetration testing with large language models [Conference paper]. Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2082-2086.

[16] Deng, G., Liu, Y., Mayoral-Vilches, V., Liu, P., Li, Y., Xu, Y., Zhang, T., Liu, Y., Pinzger, M., & Rass, S. (2024). PentestGPT: Evaluating and harnessing large language models for automated penetration testing [Conference paper]. Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24), 847-864.

[17] Isozaki, I., Shrestha, M., Console, R., & Kim, E. (2025). Towards automated penetration testing: Introducing LLM benchmark, analysis, and improvements [Conference paper]. Adjunct Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization, 404-419.

[18] Gioacchini, L., Delsanto, A., Mellia, M., Drago, I., Siracusano, G., & Bifulco, R. (2025). AutoPenBench: A vulnerability testing benchmark for generative agents [Conference paper]. Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track, 1615-1624.

[19] Skandylas, C., & Asplund, M. (2025). Automated penetration testing: Formalization and realization. Computers & Security, 155, 104454.

[20] Bailey, M., Dittrich, D., Kenneally, E., & Maughan, D. (2012). The Menlo Report. IEEE Security & Privacy, 10(2), 71-75.