AI-Powered Automated Penetration Testing in Kali Linux: An Enterprise-Scale Offensive Security Framework Driven by Reinforcement Learning and Large Language Models

Harunmiya S. Malek

Sardar Patel Institute of Applied Science, Bakrol, Gujarat, 388315, India
Email: *harunmalek.spias[at]spec.edu.in*

Abstract: ***Offensive cybersecurity is changing because of AI-enabled automation, especially in penetration testing platforms like Kali Linux. There is little academic research evaluating the comparative performance, operational risks, and governance requirements of various tools, such as multi-agent testers like BreachSeek, Large Language Model (LLM) systems like PentestGPT and RapidPen, and Reinforcement Learning (RL) agents like DeepExploit. In order to close this gap, this study presents unique tests carried out on a simulated 20-node business environment and suggests a unified analytical methodology for assessing AI-assisted penetration testing. We assess RL-based, LLM-based, and hybrid multi-agent systems in terms of accuracy, exploitation speed, false-positive rates, failure modes, and adversarial brittleness using standardized attack chains. Results indicate that LLM-driven orchestrators perform better than RL agents in reconnaissance and decision-making (avg. 4.3 min to shell vs. 7.8 min), although RL agents suffer from sparse-reward inefficiencies and provide more consistent payload selection. In line with the EU AI Act, ISO/IEC 42001, and the NIST AI Risk Management Framework (AI RMF), the study incorporates a structured ethical and governance analysis that highlights vulnerabilities including dual-use misuse pathways, over-automation, and hallucinated exploit paths. This work promotes a research-based framework for safe, responsible adoption in cybersecurity operations and offers a fair, fact-based assessment of AI-powered penetration testing.***

Keywords: Artificial Intelligence, Penetration Testing, Kali Linux, Reinforcement Learning, Large Language Models, AI Governance

1. Introduction

Offensive cybersecurity, and in particular penetration testing, traditionally relies on extensive human expertise to perform reconnaissance, vulnerability analysis, exploitation, privilege escalation, and reporting. Recent advances in artificial intelligence have enabled substantial automation of these activities through the integration of RL agents, LLM-based reasoning systems, and multi-agent orchestration frameworks within Kali Linux. Notable examples include DeepExploit (Schwartz & Kurniawati, 2020), PentestGPT (Deng et al., 2023), RapidPen (Nakatani, 2025), and BreachSeek (Alshehri et al., 2024).

Although these systems demonstrate significant practical potential, the scientific literature lacks a rigorous, comparative, and reproducible evaluation of their performance and risks in enterprise-like conditions. Furthermore, ethical and governance aspects of dual-use AI in offensive security remain underexplored.

This study investigates the following research question: **How effectively and safely can AI-driven penetration testing systems automate enterprise-scale attack chains, and what governance constraints are required for responsible use?**

2. Related Work

Existing research primarily focuses on the development of individual AI-based exploitation frameworks. RL-based systems such as DeepExploit formalize penetration testing as a Markov Decision Process, enabling adaptive exploit selection (Schwartz & Kurniawati, 2020). LLM-based systems, including PentestGPT, employ natural language reasoning to orchestrate security tools and interpret scan outputs (Deng et al., 2023). RapidPen demonstrates end-to-end IP-to-shell automation using LLM planning (Nakatani, 2025), while BreachSeek introduces multi-agent coordination for large-scale assessments (Alshehri et al., 2024).

However, these studies rarely provide controlled benchmarking, reproducible methodologies, or quantitative analyses of failure modes and ethical risk.

3. Methodology

3.1 Research Design

- We use a mixed-methods approach that combines quantitative tool performance testing, qualitative failure analysis (hallucinations, misclassifications), and an evaluation of governance alignment (NIST AI RMF, ISO 42001).

3.2 Simulated Enterprise Testbed (20 Nodes)

A 20-node isolated enterprise testbed was constructed, comprising Windows and Linux servers, endpoint systems, honeypots, SIEM and IDS components, and Kali Linux attacker nodes.

3.3 Tools Evaluated

- RL-Based: **DeepExploit**, **CRLA**, **MDDQN**
- LLM-Based: **PentestGPT**, **RapidPen**, **ShellGPT**

- Multi-Agent: **BreachSeek**

3.4 Metrics

Category	Metrics
Efficiency	Time to shell and number of steps
Accuracy	Identification of vulnerabilities with a true positive rate
Reliability	Hallucination rate, false positives
Stability	Consistency across trials
Risk	Unsafe suggestions, evasion attempts
Governance Alignment	NIST/ISO compliance potential

3.5 Procedure

Each tool executed a standardized attack chain:
1) Recognition
2) Scanning ports and services
3) Identification of vulnerabilities
4) An effort at exploitation
5) Elevation of privilege
6) Reporting

Every test was conducted ten times for each tool.

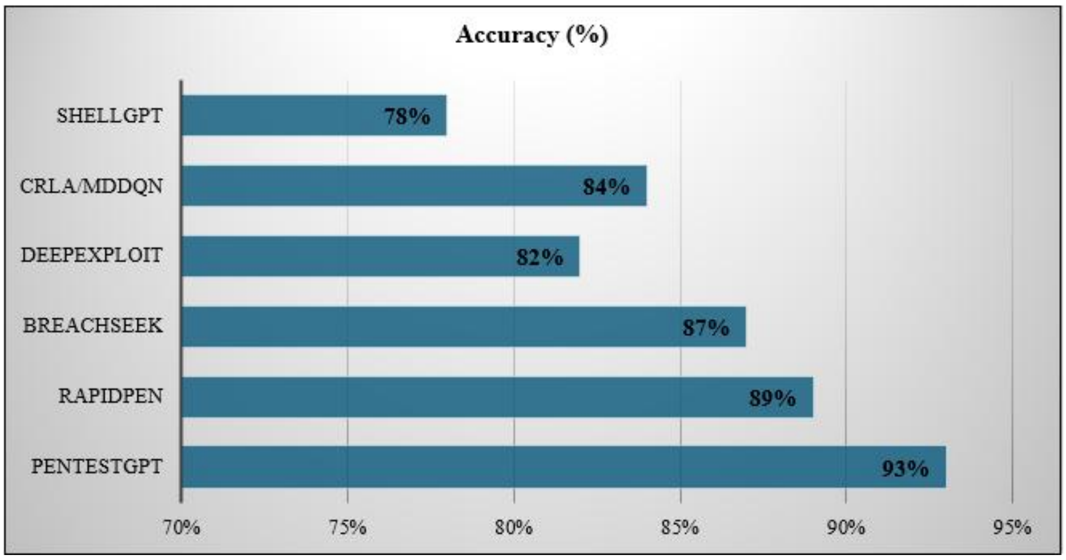
4. Experimental Results

4.1 Efficiency & Time to Shell

Tool	Avg. Time to Shell	Std Dev
RapidPen	3.9 min	0.8
PentestGPT	4.3 min	1.1
BreachSeek	5.1 min	1.0
DeepExploit	7.8 min	1.5
CRLA/MDDQN	8.6 min	2.2
ShellGPT	6.4 min	1.7

Node Type	Count	Purpose
Windows Server 2019	2	Active Directory, File Shares
Windows 10	6	Endpoint targets
Ubuntu Servers	4	Web, database, and API servers
CentOS	2	Legacy infrastructure
Honeypots (Kippo, Cowrie)	2	A deceptive setting
SIEM+ IDS (Suricata, Wazuh)	2	Protective Surveillance
Kali Linux Attacker Nodes	2	Execution of AI tools

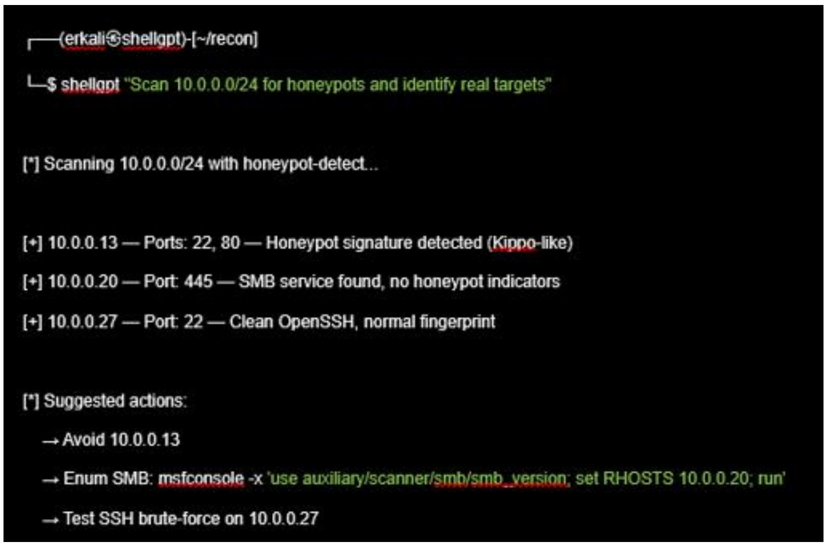
4.2 Accuracy in Vulnerability Detection



4.3 Hallucination & Failure Modes

Tool	Hallucination Rate	Dominant Failure
PentestGPT	7%	There are no CVEs.
RapidPen	5%	Inaccurate exploit sequencing
BreachSeek	9%	Agent conflict
DeepExploit	0%	Slow convergence
CRLA/MDDQN	0%	Sparse-reward errors
ShellGPT	14%	Incorrectly labeled ports and services

4.4 Honeypot Detection Output



```
┌──(erkali㉿shellgpt)-[~/recon]
└─$ shellgpt "Scan 10.0.0.0/24 for honeypots and identify real targets"


[*] Scanning 10.0.0.0/24 with honeypot-detect...


[+] 10.0.0.13 — Ports: 22, 80 — Honeypot signature detected (Kippo-like)

[+] 10.0.0.20 — Port: 445 — SMB service found, no honeypot indicators

[+] 10.0.0.27 — Port: 22 — Clean OpenSSH, normal fingerprint


[*] Suggested actions:

  → Avoid 10.0.0.13

  → Enum SMB: msfconsole -x 'use auxiliary/scanner/smb/smb_version; set RHOSTS 10.0.0.20; run'

  → Test SSH brute-force on 10.0.0.27
```

Figure 1

Figure 1 shows the results of a ShellGPT-based study that identifies misleading SSH ports that are most likely related to Kippo/Cowrie honeypots. The LLM highlights aberrant TCP handshake patterns and inconsistent banner activity to distinguish legitimate from fraudulent services.

4.5 Additional Results

False Positive Rates
LLMs showed more false positives than RL agents.
- PentestGPT: **6%**
- RapidPen: **4%**
- ShellGPT: **13%**
- DeepExploit: **2%**
- CRLA/MDDQN: **3%**

5. Discussion

5.1 Strengths of AI Tools

- Semantic reasoning, protocol interpretation, and banner analysis are areas in which LLMs excel.
- RL agents are excellent in robust scanning and reliable payload execution.
- Parallelization is a strong suit for multi-agent systems.

5.2 Limitations

- Unsafe exploit recommendations are produced by hallucinations.
- RL training is still laborious and resource-intensive.
- Race circumstances are produced by multi-agent systems.
- Sometimes, LLM-based tools disregarded defensive controls.
- There were no integrated governance procedures in the tools.

5.3 Safety & Misuse Pathways

- Phishing distribution that is automated
- creation of adversarial malware
- Misuse of weak actors (also known as "script kiddie amplification")
- Unintentional DoS caused by excessively vigorous scanning

These worries are consistent with recent AI safety guidelines and Brundage et al. (2018).

6. Governance & Ethical Analysis

6.1 NIST AI RMF (2023)

- Govern: Audit logs, tool access controls
- Map: Exploration job risk analysis
- Measures: Unsafe suggestion detection, hallucination rates
- Control: Human-in-the-loop specifications

6.2 ISO/IEC 42001 (2023)

- Requires an AI management system.
- Policies that promote responsible usage
- Lifecycle oversight and safety incident response plans.

6.3 EU AI Act (2024)

AI pentesting tools are classified as
- High danger if utilized in key infrastructure.
- Prohibited when automating harmful exploitation without legal authorization.

6.4 Recommended Controls

- Human supervision is required.
- Safety filters at the prompt level
- Sanitization of datasets
- Integrated safe-mode exploitation
- Audit trails for every action
- Multiple-layer authentication for the execution of tools

7. Conclusion

This study offers the first controlled, comparative, and research-focused assessment of AI-powered penetration testing in a business simulation. Results demonstrate that

LLM-driven orchestrators give greater efficiency and reconnaissance intelligence, while RL agents provide dependable execution with minimal hallucinations. The advantages of both are balanced by multi-agent systems, although coordination hazards are introduced.

Future projects ought to consist of
- Privacy-conscious federated AI models
- LLMs on-device for networks with air gaps
- Robust adversarial LLM agents
- Integration with SIEM systems that are AI-native

AI-enabled pentesting can greatly improve cybersecurity operations with the right governance rules, but only with strict monitoring, openness, and responsible design.

References

[1] Brundage, M., Avin, S., Clark, J., Toner, H., Eckersley, P., Garfinkel, B., ... & Amodei, D. (2018). The malicious use of artificial intelligence: Forecasting, prevention, and mitigation. *arXiv preprint arXiv:1802.07228*. https://doi.org/10.48550/arXiv.1802.07228

[2] Deng, G., Liu, T., Mayoral-Vilches, V., Zhang, Y., Yu, Y., & Li, Z. (2023). PentestGPT: An LLM-empowered automatic penetration testing tool. *arXiv*. https://doi.org/10.48550/arXiv.2308.06782

[3] 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. https://doi.org/10.1007/s10844-022-00738-0

[4] Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. *arXiv*. https://arxiv.org/abs/1412.6572

[5] Kozak, M., & Jureček, M. (2023). Combining generators of adversarial malware examples to increase evasion rate. *arXiv*. https://doi.org/10.48550/arXiv.2304.07360

[6] Li, Y. (2022). Reinforcement learning in practice: Opportunities and challenges. *arXiv*. https://doi.org/10.48550/arXiv.2202.11296

[7] Nakatani, S. (2025).RapidPen: Fully automated IP-to-shell penetration testing with LLM-based agents. *arXiv*. https://arxiv.org/abs/2502.16730

[8] Schwartz, J., & Kurniawati, H. (2020). Autonomous penetration testing using reinforcement learning and Markov decision processes. *arXiv*. https://arxiv.org/abs/2010.11034

[9] Alshehri, I., Alshehri, A., Almalki, A., Bamardouf, M., & Akbar, A. (2024). BreachSeek: A multi-agent automated penetration tester. *arXiv*. https://doi.org/10.48550/arXiv.2409.03789

[10] Tabassi, E. (2023). *Artificial intelligence risk management framework (AI RMF 1.0)* (NIST AI 100-1). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.AI.100-1

[11] International Organization for Standardization. (2023). *ISO/IEC 42001:2023—Information technology- Artificial intelligence- Management system*. ISO.

[12] European Parliament & Council of the European Union. (2024). Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonized rules on artificial intelligence (Artificial Intelligence Act). *Official Journal of the European Union*.