LLM Agents

Menashe Ashkenazi[1]

[1]Computer Science, Ariel University, Israel, `menashe.Ashkenaz@msmail.ariel.ac.il`

1. Introduction

The increasing complexity of digital infrastructure and the rapid evolution of cyber threats have rendered traditional penetration testing insufficient in many real-world scenarios. Manual and semi-automated methods are no longer scalable in environments that require constant monitoring, rapid decision-making, and dynamic adaptation to new vulnerabilities.

Recent advances in artificial intelligence (AI), particularly in deep reinforcement learning (DRL) and large language models (LLMs), offer a powerful opportunity to redefine the field of automated cybersecurity. These methods enable the design of intelligent agents that can not only detect, but also autonomously act upon discovered vulnerabilities in a simulated or real attack environment.

In this work, we propose an agent-based AI system that simulates autonomous cyber-attacks, structured around a hierarchy of task-specific agents. Each agent specializes in a different stage of the cyber kill chain—reconnaissance, vulnerability identification, exploitation, and post-exploitation—and learns optimal behavior through continuous interaction with its environment. The system integrates deep reinforcement learning for decision-making and LLMs for interpreting noisy command-line outputs in real time.

Our goal is to build a generalizable and extensible framework in which agents collaborate, adapt, and improve over time. Unlike rule-based systems, our architecture allows the agents to learn from previous actions, share experiences via a centralized replay buffer, and even generate specialized sub-agents on demand.

2. Motivation and Objectives

Although artificial intelligence has made impressive strides in classification, prediction, and natural language understanding, its role in dynamic decision-making during cyberattacks remains underexplored. Most AI applications in cybersecurity are defensive—focused on anomaly detection or threat prediction. In contrast, this research focuses on the offensive side: how AI can plan, adapt, and execute cyberattacks in evolving and uncertain environments.

The motivation stems from the inherent unpredictability of real-world systems. A single IP address may expose different services depending on time, configuration, or user behavior. No fixed rule-set or pre-scripted logic can handle all possible scenarios. Thus, there is a need for an autonomous system that learns from interaction, identifies patterns, and refines its tactics over time.

Furthermore, many tasks required in offensive operations involve **interpreting raw, noisy, or partially structured output** from network commands. While humans can easily infer useful information from such data, traditional parsers and scripts fail when faced with slight format deviations or unexpected content. Large language models (LLMs) offer a compelling solution, but they must be guided and embedded within a learning agent framework to be reliable and consistent.

Based on these motivations, this project aims to achieve the following objectives:

1. **Agent-Driven Design:** Build a hierarchy of agents, each with a specific role (e.g., reconnaissance, vulnerability lookup, exploitation), coordinated by a central management layer.

2. **Learning by Interaction:** Implement deep reinforcement learning mechanisms for each agent to optimize behavior through trial-and-error across multiple attack episodes.

3. **LLM-Augmented Perception:** Integrate large language models to process command-line output and convert it into structured internal representations suitable for decision-making.

4. **Meta-Level Adaptation:** Develop an agent responsible for high-level strategy, determining which agents should act and in what sequence, based on historical performance.

5. **Self-Extension:** Enable the system to autonomously clone and fine-tune new agents for specific technologies (e.g., Apache, FTP), thus specializing over time.

3. Existing Datasets and Resources

The proposed system builds upon multiple open-source cybersecurity datasets and real-world resources, allowing the agents to operate in realistic and complex environments. These resources provide ground truth for vulnerability data, exploit metadata, and command-line behavior during attacks.

The key datasets and resources used in this project include:

- **National Vulnerability Database (NVD):** A JSON-based repository of CVE records maintained by NIST, providing standardized descriptions of software vulnerabilities, CVSS severity scores, and mappings to affected products via the CPE taxonomy National Institute of Standards and Technology (NIST), n.d.
- **ExploitDB:** A public collection of curated exploits and proof-of-concept code, indexed by CVE identifier and categorized by platform, attack type, and vulnerability class. This dataset enables linking vulnerabilities to real-world attack implementations Offensive Security, n.d.
- **Metasploitable2 Virtual Machine:** A deliberately vulnerable Linux-based virtual machine designed for testing penetration tools. It includes exploitable services such as FTP, SSH, Apache, WebDAV, phpMyAdmin, and DVWA, and serves as the simulation target for training autonomous agents Rapid7, 2012.
- **State-Action-Replay Logs:** Automatically collected during training episodes, each log includes the full state, action taken, the result and the quality of the attack. This dataset supports continual reinforcement learning.

4. Related Work and Literature Review

Several recent studies have investigated the use of artificial intelligence in the context of penetration testing, agent-based decision-making, and cyber-defense architectures. However, few have explored the full integration of hierarchical reasoning, meta-learning, and reflective self-analysis within autonomous offensive systems.

Hu et al. (2020) Hu et al., 2020 introduced a pioneering approach using deep reinforcement learning (DRL) to automate penetration testing in simplified simulated networks. Their work demonstrated that DRL agents could learn attack sequences through exploration and reward signals. However, their design was limited to monolithic agents, lacked modularity, and did not incorporate real-world tools or dynamic parsing of output.

Ghanem et al. (2023) Ghanem et al., 2023 extended this idea by proposing a hierarchical reinforcement learning framework specifically tailored to large-scale networks. Their model decomposes attack paths into subtasks across subnetworks, significantly improving learning efficiency and scalability. While this hierarchical breakdown aligns with our agent-based modular design, their system does not include real-time language processing or blackboard-based memory for multi-agent coordination.

Ahmad et al. (2023) Ahmad et al., 2023 applied meta-learning techniques to cyber-attack detection in IoT networks. Their meta-learner aggregates predictions from deep classifiers (CNN, RNN, LSTM) to generalize across unseen attack types. While their work focused on anomaly detection rather than active exploitation, it underscores the value of meta-learning in security contexts—particularly for systems facing diverse and evolving threats.

Cai et al. (2025) Cai et al., 2025 introduced AegisLLM, a novel self-reflective framework for defending large language models (LLMs). Their system is built around a modular agentic architecture composed of *specialized agents*, each responsible for a distinct aspect of defense: the **Orchestrator** manages strategic planning and agent delegation, the **Deflector** handles adversarial prompts and threat mitigation, and the **Evaluator** continuously monitors outputs to detect inconsistencies or failures. These agents operate in a tightly coordinated loop, enabling real-time self-reflection and corrective behavior. While AegisLLM is designed for defense rather than offensive penetration testing, its use of explicit agent specialization, modular coordination, and runtime self-reflection strongly influences the architectural direction of our proposed offensive framework.

Research Gap: Despite these advances, no existing work combines:

- Hierarchical reinforcement learning agents for modular cyberattacks,
- Meta-reasoning mechanisms to manage agent selection and adaptation,
- Self-reflective agent orchestration enabling dynamic creation and delegation of specialized agents, based on task context, past performance, and runtime feedback.
- LLM-powered output parsing for real-world command-line environments.

Our work aims to bridge this gap by building a fully autonomous, modular penetration testing system, governed by a meta-agent capable of strategic planning, agent delegation, and enabling dynamic agent creation and reflective self-analysis throughout the attack lifecycle.

5. Proposed Solution and Agent Architecture

5.1 Overall System Overview

We propose a fully autonomous penetration testing framework that integrates deep reinforcement learning, large language models (LLMs), and agent-based reasoning. The system operates in a loop of exploration, interpretation, decision-making, and learning, guided by a centralized blackboard memory shared across all agents. Each agent specializes in a specific class of actions (e.g., reconnaissance, exploitation, privilege escalation), and the entire system is coordinated by a meta-manager agent responsible for high-level control and delegation. The framework is designed to adapt dynamically to unknown environments by learning from past interactions and adjusting agent behavior in real time.

5.2 Hierarchy of Agents

The architecture follows a hierarchical design in which agents are organized into layers based on their level of abstraction and scope of control. At the base level are action-specific agents (e.g., `ReconAgent`, `AccessAgent`) that interact directly with the target system through command-line tools. Mid-level agents handle decision logic, evaluate results, and communicate updates to the shared blackboard. At the top of the hierarchy, the `MetaManager` agent monitors progress, evaluates outcomes, and adjusts agent priorities or assignments as needed. This hierarchy enables scalable coordination, parallelism, and knowledge flow from low-level interactions to high-level strategic planning.

5.3 The Role of the Meta-Manager

The `MetaManager` agent serves as the cognitive control unit of the system. It is trained using reinforcement learning and meta-learning techniques to learn optimal policies for agent selection, delegation, and sequencing. Beyond static coordination, it also performs self-evaluation based on global system feedback, blackboard metrics, and reward histories. Its responsibilities include:

- Selecting which agent to activate at each time step.
- Modifying the agent pool by spawning specialized clones for observed targets.
- Updating high-level strategies based on task success/failure.
- Reflecting on previous decisions to improve future delegation.

5.4 Agent Specialization and Dynamic Cloning

Each agent is designed to learn a distinct behavior policy suited for a specific function in the attack process. As the system encounters repeated targets (e.g., a known Apache service), the meta-manager can trigger the creation of a specialized clone of an existing agent, fine-tuned for that scenario. These clones are added to an *agent zoo* — a dynamic repository of trained specialists. During execution, the meta-manager queries this zoo to select the most appropriate agent or instantiate a new one if needed. This mechanism enables few-shot adaptation, modular transfer learning, and long-term accumulation of expertise across different environments.

6. Subsystems, Capabilities and Supporting Modules

6.1 LLMParserAgent and Output Interpretation

6.2 Blackboard and State Tracking

6.3 Reward Functions and Learning Feedback

6.4 Agent Zoo and Experience Replay Sharing

7. Deep Learning Models

7.1 General Architecture of Agent Policy Networks

7.2 Training Objectives and Algorithms

7.3 Custom Deep Learning Model (Developed From Scratch)

7.4 LLM Fine-Tuning for Structured Output

7.5 Integration of Open-Source LLMs (LLaMA, Mistral)

8. Implementation Plan and Timeline

9. Expected Outcomes and Evaluation Metrics

10. Conclusion

References

Ahmad, W., Azam, M. A., Bangyal, W. H., Almogren, A., & Akhunzada, A. (2023). Meta-learner-based approach for detecting attacks on internet of things networks. *Sensors*, 23(17), 7592. https://doi.org/10.3390/s23177592

Cai, Z., Shabihi, S., An, B., Che, Z., Bartoldson, B. R., Kailkhura, B., Goldstein, T., & Huang, F. (2025). Scaling agentic systems for self-reflective defense in llm security [Available at https://openreview.net/forum?id=HtJ75I6KDG]. *OpenReview (preprint)*.

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*. https://doi.org/10.1007/s10844-023-00755-1

Hu, H., Yang, Y., Qian, C., Yu, T., & Zhang, B.-T. (2020). Automated penetration testing using deep reinforcement learning. *IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 112–121. https://doi.org/10.1109/EuroSPW51379.2020.00025

National Institute of Standards and Technology (NIST). (n.d.). National vulnerability database (nvd).

Offensive Security. (n.d.). Exploit database (exploitdb).

Rapid7. (2012). Metasploitable2 - vulnerable virtual machine.