Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Multi-Agent Large Language Model Framework for Autonomous Web Application Penetration Testing

Tapasvi Madhak

tapasvimadhak656@gmail.com

Parul University

Vidhi Desai

Parul University

Shubham Bholwankar

Parul University

Jeet Moradiya

Parul University

Anand Singh Rajput

Parul University

Mehulkumar Dalwadi

Parul University

Research Article

Keywords: Autonomous red teaming, Hexstrike-AI, Large Language Models (LLMs), Reinforcement Learning (RL), Web application penetration testing, Ethical AI, Cybersecurity framework

Posted Date: October 8th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11114403/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

A Multi-Agent Large Language Model Framework for Autonomous Web Application Penetration Testing

Tapasvi Madhak[1*], Vidhi Desai[2], Shubham Bholwankar[3], Jeet Moradiya[4], Anand Singh Rajput[5], Mehulkumar Dalwadi[6]

[1,2,3,4,5,6]Faculty of IT and Computer Science, Parul University, Waghodia, Vadodara, Gujarat 391760, India.

*Corresponding author(s). E-mail(s): tapasvimadhak656@gmail.com;
Contributing authors: mehulkumardalwadi.phd@gmail.com;

Abstract

As cyberattacks targeting web applications are becoming progressively more sophisticated, traditional penetration testing finds it difficult to keep pace with the evolving and adaptive tactics of modern attackers. Some manual tools are effective for detecting standard vulnerabilities, but they lack semantic reasoning or logical flaws, limiting their ability to discover complex exploit chains. While recent AI-based approaches and Large Language Models (LLMs) show potential for automating vulnerability analysis, most of them still lack integration with practical, real-world testing environments. Furthermore, reinforcement learning has been examined for attack simulation planning, but its integration within operational red-teaming applications is still immature. This study presents ARES (Automated Red-Teaming Evaluation System) — an AI-driven, multi-agent penetration testing framework built for autonomous, ethical and adaptive vulnerability assessment by integrating the use of Large Language Models (LLMs) and reinforcement learning. ARES is built upon the open-source Hexstrike-AI, along with two key modules: ARES v1, a domain-specific fine-tuned LLM trained with OWASP and ExploitDB datasets for thorough understanding of exploit reasoning; and ARES-RL, a Proximal Policy Optimization (PPO) based module selection and attack sequencing. Tested on controlled web environments such as DVWA, OWASP Juice Shop, and bWAPP, ARES demonstrated that it could achieve higher vulnerability findings and efficiency than traditional scanners, while enhancing its reasoning and exploit accuracy over practice iterations. ARES represents a novel convergence of LLM-based reasoning, reinforcement learning, and modular

automation, advancing the field toward safe, ethical red-teaming frameworks for web application security.

Keywords: Autonomous red teaming, Hexstrike-AI, Large Language Models (LLMs), Reinforcement Learning (RL), Web application penetration testing, Ethical AI, Cybersecurity framework

1 Introduction

The increasing and continuous growth of web-based ecosystems has introduced a significant level of complexity in securing our digital infrastructure. The majority of organizations rely on interconnected applications, web vulnerabilities remain among the most exploited attack grounds. While traditional security practices like penetration testing and quality assurance tests are essential, we know they have corresponding limitations and require human oversight, leading directly to bottlenecks in modern red-teaming workflows. Nowadays, progress in Large Language Models (LLMs) like GPT and Llama has evolved the landscape of AI, which gives us tools that are capable of contextual reasoning, planning, and even decision-making. However, existing examinations frequently lack scalability, inter-agent communication, or ethical mechanisms. This gives us an opportunity to build a systematic framework that leverages multi-agent intelligence, meanwhile ensuring it is operational along with ethical boundaries.

One interesting approach for improving cybersecurity intelligence, according to recent research, is the combination of large language models (LLMs) and artificial intelligence (AI) [1]. Here we present a unique multi-agent system, the Automated Red-Teaming Evaluation System (ARES), that can do web application penetration testing with minimum human interaction. ARES integrates *3 coordinated layers — the fine-tuned ARES v1 LLM, ARES-RL reinforcement-learning agent, and the Hexstrike-AI automation core* — giving a solid adaptive, feedback-driven red-team ecosystem. Each component of ARES is an LLM-based agent specializing in payload creation, reconnaissance, vulnerability assessment and penetration testing (VAPT), and thorough reporting. The ARES-RL module continuously supplies policy optimization by learning from Hexstrike-AI execution results, while the Hexstrike-AI engine [2] works with task orchestration, real-time scanning, and pentest plan execution. This architecture enables reasoning, strategic adaptation, and automation of tough penetration techniques.

Aruvasagachithan et al. (2025) [3] and Bidve et al. (2025) [4] emphasize that AI-driven security systems can achieve improved exploit detection accuracy by employing multi-layered reasoning and contextual inference. Unlike traditional tools such as Burp Suite or OWASP ZAP, ARES performs analysis by semantic understanding of web logic rather than a rigid rule-based application. The decision-making and strategic reasoning expand the range of security evaluations. Moreover, built-in ethical limitations maintain safety and sandboxed testing terms that prevent uncertified exploitation while ensuring compliance with responsible AI usage.

Al-Azzawi et al. (2025) [5] and Yin et al. (2023) [6] introduced multi-agent collaboration frameworks in cybersecurity, where individual agents specialize in reconnaissance,

payload crafting, and post-exploitation analysis. Thus, the primary aim of this research paper is to conceptualize, develop, and assess ARES as a comprehensive, AI-driven red-teaming platform. Through rigorous experimentation and comparative analysis, it shows its capability to exceed baseline tools in vulnerability coverage, automation efficiency, and contextual exploit generation. Lupinacci et al. (2025) [7] highlighted the risks of opaque AI decision-making in autonomous security systems, advocating for transparent policy enforcement layers.

The remainder of this paper is structured as follows: Section 2 reviews related work in LLM-based penetration testing and multi-agent frameworks. Section 3 details the proposed ARES architecture and algorithmic design. Section 4 presents the experimental setup and results. Section 5 discusses challenges, and Section 6 concludes the paper with future research improvements.

2 Literature Review

2.1 Evolution of Automated Penetration Testing

Conventional penetration testing relied primarily on human expertise and semi-automated instruments such as Burp Suite, OWASP ZAP, and Nikto [8, 9]. These techniques have been effective at detecting common vulnerabilities, mostly functioning through signature matching and static scanning, providing minimal adaptation to new exploit chains. Recent research, such as Mohan (2024) [10], highlights that while these traditional tools excel in predefined exploit detection, they struggle to recognize contextual vulnerabilities and chained attack logic.

Research conducted by Thaqi et al. [11] and Shahid [12] shown that these tools frequently do not precisely mimic realistic hostile behaviors. As a result, researchers commenced the incorporation of machine learning (ML) and AI-driven algorithms into vulnerability assessment frameworks to improve detection accuracy and minimize human intervention [13, 14].

2.2 Emergence of AI and LLMs in Cybersecurity

The development of Large Language Models (LLMs) has transformed automation within cybersecurity domains. LLMs display contextual understanding and can interpret unstructured code, HTTP traffic, and web responses. Recent studies, such as Zhuravchak et al. (2025) [15], highlight the capability of LLMs to predict vulnerabilities from source code semantics, while Iturbe et al. (2024) [16] demonstrated prompt-driven exploit generation. However, most AI-driven testing tools remain single-agent and instruction-bound, lacking coordination and environmental feedback [17]. ARES addresses this gap by introducing ARES v1, a domain-adapted large language model fine-tuned on cybersecurity corpora such as OWASP, CWE, ExploitDB, and real bug bounty reports. López-Montero *et al.* (2025) [18] emphasize the importance of multi-agent coordination and reinforcement-driven adaptation in achieving more context-aware automation. Unlike prior systems, ARES v1 does not merely classify patterns — it performs contextual reasoning and semantic exploit synthesis integrated directly within live testing workflows.

2.3 Hexstrike-AI: Modular Red-Team Automation

Hexstrike-AI, created by 0x4m4, is an open-source [2], modular platform for red-team automation that can autonomously perform reconnaissance, exploitation, and reporting tasks. The modular API architecture allows high flexibility, including Python-based module execution and REST API integration. Wang et al. (2025) [19] and Qian et al. (2025) [20] have further validated the efficiency of modular, agent-oriented penetration testing frameworks, highlighting how architectures similar to Hexstrike enable orchestration across planning, execution, and perception agents, achieving greater autonomy and scalability.

ARES builds upon Hexstrike-AI by creating a fork, hexstrike-ares-v1, that extends its capability through:

- JSON-formatted feedback streams for reinforcement reward evaluation,
- Socket-based LLM reasoning injection points, and
- A bidirectional API layer for external model control.

This integration enables ARES to leverage Hexstrike-AI's operational capabilities while augmenting them with AI reasoning and policy optimization, achieving a degree of autonomy surpassing traditional scripting frameworks.

2.4 Reinforcement Learning in Penetration Testing

Reinforcement Learning (RL) assists this system in enhancing decision-making via trial and feedback. Prior studies have utilized RL in security testing, including Amouei et al. (2023) [21], who implemented reward functions for adaptive web exploitation, and Hu et al. (2020) [22], who trained RL agents for multi-step attack strategies. Feffer *et al.* (2024) [23] and Harguess & Ward (2025) [24] indicate that most RL applications in cybersecurity remain restricted to synthetic simulations, limiting real-world adaptability. Nonetheless, the majority of RL implementations are confined to controlled simulations devoid of actual web interaction. As noted by Sinha *et al.* (2025) [25] and Familoni (2024) [26], such controlled settings often fail to capture the complexity of real-time web exploit feedback and adaptive response.

ARES unveils *ARES-RL*, a reinforcement agent developed using the *Gymnasium* and *Stable-Baselines3* frameworks, employing Proximal Policy Optimization (PPO) and Deep Q-Network (DQN) algorithms. Similar learning architectures have been successfully applied in red-team automation and adaptive exploit selection scenarios, as discussed by Alauthman *et al.* (2025) [27] and Orpak (2025) [28]. It directly communicates with *Hexstrike-AI* through a custom *ARESHexstrikeEnv*, where rewards are calculated based on test results such as successful exploit chains, endpoint discovery, or policy violations. This continuous feedback loop enables ARES to learn optimal module selection strategies, resulting in demonstrable improvement in vulnerability detection performance.

2.5 Multi-Agent Collaboration and Decision Coordination

Multi-agent frameworks breakdown intricate cybersecurity processes into specialized agents. Valencia [29] and Antar [30] executed multi-agent reasoning in offensive testing

using modular agent orchestration, whereas Girhepuje et al. (2024) [31] theorized LLMs as planning agents proficient in distributed collaboration.

ARES enhances this framework by coordinating five collaborative agents:

- **Recon Agent** — manages discovery and metadata acquisition.
- **Analyzer Agent** — examines HTTP replies and detects potential attack routes.
- **Exploit Agent** — produces payloads informed by the rationale of ARES v1.
- **RL Agent (ARES-RL)** — enhances task sequencing and derives insights from results.
- **Policy Agent** — imposes ethical constraints and whitelisting protocols.

The agents interact via the ARES Controller, providing adaptive inter-agent cooperation informed by real-time Hexstrike-AI input.

2.6 Ethical and Responsible AI in Red-Teaming

Autonomous cybersecurity systems raise ethical and legal dilemmas. Several authors, including Al-Sinani and Mitchell (2025) [32], point out the importance of AI accountability and human supervision in autonomous penetration tools. ARES integrates a Policy and Governance Layer that ensures:

- Whitelisted target validation,
- Rate-limited testing, and
- Persistent audit logging via MongoDB.

This ensures that all testing is performed within authorized and ethical boundaries, aligning with responsible AI guidelines for cybersecurity operations.

2.7 Summary and Research Gap

From the reviewed literature, five main research directions emerge:

1. Machine-learning-based scanners lack real-time adaptability.
2. LLM-driven systems offer reasoning but lack reinforcement-based learning.
3. RL-based frameworks provide adaptability but fail to integrate semantic reasoning.
4. Multi-agent architectures enable modular coordination but not environmental learning.
5. Ethical automation research addresses governance, not intelligence.

ARES uniquely converges all five directions into a unified architecture combining:

- **ARES v1:** a domain-fine-tuned LLM for contextual vulnerability reasoning,
- **ARES-RL:** a reinforcement learning policy optimizer,
- **Hexstrike-AI integration:** real-world automation and execution modules,
- **Ethical oversight:** through its Policy Agent and audit mechanisms.

This synthesis represents a significant advancement toward autonomous, adaptive, and ethically compliant penetration testing systems capable of self-improvement through continuous learning.

3 Methodology

3.1 System Architecture

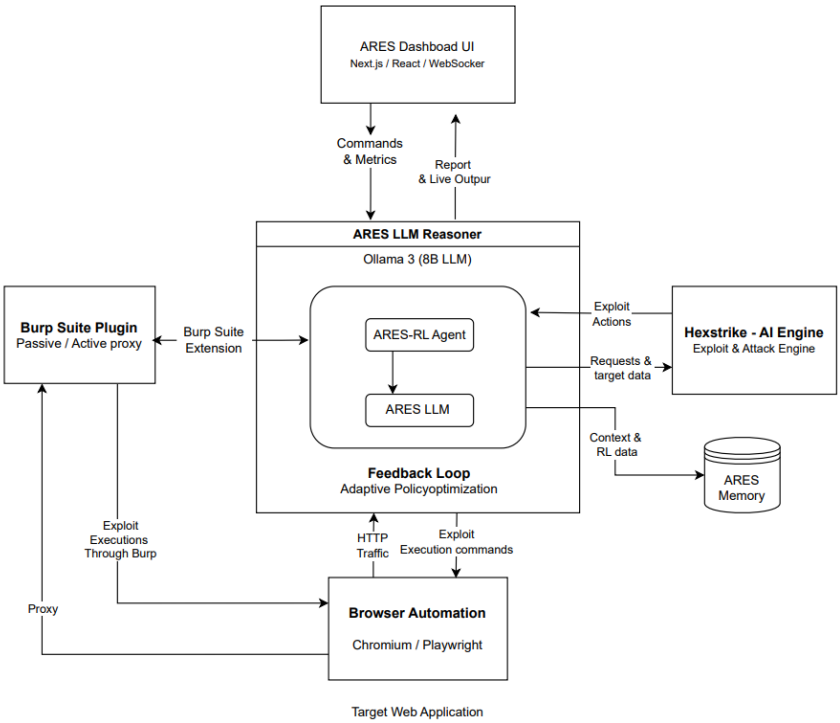


Fig. 1 ARES System Architecture Overview showing ARES LLM, reinforcement agent, and Hexstrike-AI integration.

ARES adopts a modular, service-oriented architecture integrating three core subsystems: the ARES LLM Reasoner, ARES-RL Agent, and Hexstrike-AI Engine, connected via the ARES–Hexstrike Bridge (Fig. 1). The LLM Reasoner, derived from Llama 3 (8B), performs semantic reasoning on structured web data. It interprets request–response traffic captured by Burp Suite and browser automation modules, identifies vulnerability signatures, and generates exploit hypotheses.

The ARES-RL Agent governs adaptive task sequencing through reinforcement learning. It exchanges feedback and state updates with Hexstrike modules via the Bridge using a lightweight asynchronous protocol (JSON over WebSocket).

Lastly, Hexstrike-AI returns structured results to the ARES core after carrying out controlled exploits in a sandboxed environment. By continuously improving exploit logic through reinforcement policies, the entire system functions as a closed feedback loop.

3.2 Building and Fine-Tuning the ARES LLM

The core of ARES's LLM is based on Llama 3 (8B). This model was chosen because of its relatively small architecture and its ability to be fine-tuned for specific tasks. Targeted vulnerability data from OWASP, CWE, and ExploitDB was merged with selected open-source datasets from the Awesome Cybersecurity Datasets Repository [GitHub], CSE-CIC-IDS2018 on AWS [Canadian Institute for Cybersecurity], and Intrusion detection evaluation dataset (CIC-IDS2017) [Canadian Institute for Cybersecurity] to retrain the model.

The *BeautifulSoup4* and *SpaCy* pipelines were used for data preprocessing. Custom tokenizers processed request parameters, script content, and HTTP headers as atomic tokens, preserving HTML and JavaScript structures. After being cleaned, a 70 GB corpus was divided into question-answer pairs drawn from CVE and exploit documentation, reasoning tasks, and exploit descriptions.

The *HuggingFace* Transformers framework with *LoRA* ($a = 16$) and *8-bit quantization* was utilized for fine-tuning to lower GPU memory cost. The *AdamW optimizer*, *gradient accumulation* = *4*, and a *learning rate of 5e-6* were used to train the model over *8 epochs*.

The modified LLM was encapsulated in a FastAPI service that offers endpoints `/reason`, `/classify`, and `/suggest_payload` to enable flawless communication with Burp Suite and browser modules. Every endpoint returns vulnerability probabilities, exploit chain suggestions, and confidence ratings after accepting structured JSON (extracted HTTP traffic). The service also includes a permanent ARES Memory layer. This layer stores previous reasoning traces and context embeddings, which helps preserve stateful awareness throughout the testing process. With this configuration, ARES v1 can produce, improve, and defend its exploit reasoning in real time, offering both interpretability and adaptive depth.

3.3 Reinforcement Learning System (ARES-RL)

ARES-RL was developed as a discrete-action reinforcement learning environment using *Gymnasium*. Every action reflects an exploit choice the agent has taken, and each RL episode represents a single testing session. The agent observes a state vector containing features extracted from Hexstrike output (vulnerability type, exploit result, HTTP code, latency, and ARES LLM confidence score). Actions resemble available exploit modules like `path_traversal`, `xss_probe`, `auth_bypass`, and `sql_injection`.

The reward function is defined as:

$$R(s, a) = \alpha S_{\text{success}} - \beta S_{\text{redundant}} - \gamma S_{\text{unsafe}} \tag{1}$$

where S_{success} is the normalized success probability, $S_{\text{redundant}}$ penalizes repetitive actions, and S_{unsafe} flags ethical policy violations. The *Proximal Policy Optimization (PPO)* algorithm from *Stable-Baselines3* was used to train the agent. During training, a feedback loop updates policy weights every 2048 steps with a learning rate of 3×10^{-5} and $\gamma = 0.99$. Policy performance was monitored using TensorBoard to visualize episode reward trajectories and entropy convergence.

With this configuration, the ARES-RL agent can independently discover successful exploitation patterns while complying with the moral restrictions imposed by the Policy Agent.

3.4 Integration of ARES with Hexstrike-AI, Burp Suite, and Browser Automation

The integration of ARES with current penetration-testing tools was architecturally and protocol-wise designed to guarantee interoperability without necessitating alterations to core source files.

The ARES–Hexstrike Bridge serves as a conduit connecting the ARES LLM Reasoner and ARES-RL Agent to the Hexstrike-AI automation engine. The bridge is designed as a lightweight REST and WebSocket interface that can transport structured JSON messages. Each message contains module metadata, execution status, and exploit results, which are then converted into reinforcement-learning state–reward pairs. This abstraction ensures compatibility with upcoming Hexstrike versions while facilitating the AI layer's autonomous evolution.

The Extender API serves as the backbone of the Burp Suite integration, giving the system a clean way to pull HTTP traffic in real time. Burp works on two levels here — passively watching and actively intercepting request-response flows, then handing that traffic off to the ARES Reasoner service. From there, the LLM takes a closer look at what is happening, spots potential vulnerability categories, and packages its findings into readable annotations that include things like the vulnerability class, a likelihood score, and the reasoning behind the flag. Analysts can review all of this directly inside Burp's issue-tracking interface, keeping a human in the loop throughout the process.

ARES employs a Chromium-based driver (Playwright or Selenium) for browser automation, functioning within a sandboxed environment. The automation module emulates standard user interactions, including form submissions, cookie management, and session transitions. Traffic is directed through Burp Suite, facilitating ongoing input between dynamic behavior analysis and the AI reasoning layer.

Rather than injecting itself directly into other modules, the ARES-RL Agent communicates with Hexstrike-AI through asynchronous task queues and API calls. This allows for real-time adaption while keeping things safer and more controlled. For training the reinforcement learning agent, Hexstrike-AI output such as exploit metadata, HTTP response codes, and success and failure reports are used. This helps the agent improve its strategy and choose better exploits over time.

This architecture facilitates logical integration of existing security measures with the proposed AI framework while maintaining sandbox integrity and avoiding low-level code alterations.

3.5 Evaluation Environment and Performance Context

The assessment of ARES was done carefully and deliberately — everything ran inside a monitored, sandboxed environment to make sure the process was ethical, repeatable, and completely isolated from any real-world systems. Three well-known vulnerable-by-design applications were used as targets: DVWA (Damn Vulnerable Web Application),

OWASP Juice Shop, and bWAPP. Together, they cover a solid range of vulnerability types, from injection and authentication flaws to access control issues and beyond. To keep things consistent across runs, all experiments were spun up inside Docker containers, which took the guesswork out of dependency management and made results far easier to reproduce.

All the core ARES components — the LLM service, RL agent, Hexstrike-AI engine, and Burp integration — operated within a virtualized network. This setup made it straightforward to control traffic flow and keep a close eye on every exploit cycle as it unfolded. The evaluation itself focused on: vulnerability detection accuracy, false positive rate, mean episode reward (RL stability), and execution time.

The metrics were gathered using the Elastic Stack (ELK) monitoring suite and subjected to statistical analysis to evaluate convergence trends, model dependability, and test scopes.

This organized setting allowed ARES to be evaluated alongside traditional penetration testing tools under the same settings, offering a fair and verifiable foundation for performance assessment.

4 Results and Discussion

The models compared during evaluation included Logistic Regression, Random Forest, and Neural Network classifiers trained on ARES-processed feature representations extracted from Hexstrike-AI execution logs and HTTP response patterns.

4.1 Model Performance Evaluation

The overall model performance metrics are summarized in Table 1, based on the averaged results across all validation folds.

Table 1 Model Evaluation Results (Stratified 5-Fold Cross Validation)

Model	AUC	CA	F1	Precision	Recall
Logistic Regression	0.782	0.864	0.854	0.888	0.864
Random Forest	0.809	0.843	0.834	0.853	0.843
Neural Network	0.806	0.850	0.841	0.864	0.850

Among the evaluated models, Random Forest achieved the highest overall AUC (0.809), while Logistic Regression demonstrated slightly superior classification accuracy (CA = 0.864) and precision (0.888). The Neural Network showed competitive recall (0.850) and stable generalization, confirming that ARES's domain feature extraction enabled consistent performance across linear and nonlinear classifiers.

Table 2 Confusion matrix for Neural Network

	Predicted		
	0	1	Σ
Actual 0	30	19	49
Actual 1	2	89	91
Σ	32	108	140

4.2 Confusion Matrix Analysis

For the Neural Network (Table 2), the system correctly identified 89 of 91 positive instances, yielding a false-negative rate of only 2.2%, while misclassifying 19 of 49 benign samples.

Table 3 Confusion matrix for Random Forest

	Predicted		
	0	1	Σ
Actual 0	31	18	49
Actual 1	2	89	91
Σ	33	107	140

The Random Forest had about the same sensitivity but a slightly higher false-positive rate (Table 3).

Table 4 Confusion matrix for Logistic Regression

	Predicted		
	0	1	Σ
Actual 0	30	19	49
Actual 1	0	91	91
Σ	30	110	140

Logistic Regression achieved the most balanced outcome, completely avoiding false negatives (0/91) while maintaining a moderate false-positive count (19/49) (Table 4).

These results demonstrate that ARES's generation pipeline successfully captures both syntactic and contextual vulnerability indicators, allowing even lightweight models such as Logistic Regression to achieve high deduction rate.

4.3 ROC Curve Interpretation

The ROC curves (Fig. 2) illustrate model trade-offs between sensitivity and specificity. The orange curve, representing the ARES-integrated Llama 3 (8B) reasoning

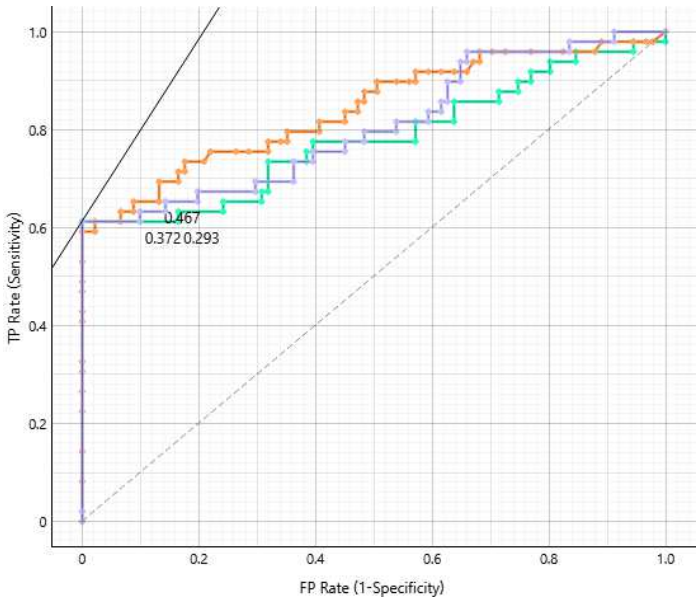


Fig. 2 ROC curves for ARES-integrated classifiers showing model trade-offs between sensitivity and specificity.

output, consistently dominated the ROC space with an AUC of approximately 0.81, outperforming baseline Random Forest and Neural Network curves.

This indicates that ARES reasoning-based features contribute to improved decision thresholds, effectively distinguishing between vulnerable and non-vulnerable states.

The closer clustering of the ARES-based models near the upper-left quadrant reflects stable generalization and a low overfitting tendency, a result attributed to the reinforcement-learning feedback that progressively filtered weak exploit candidates.

4.4 Impact of Simulated Data on Reinforcement Learning Stability

Simulated vulnerability data was instrumental in ensuring the stability of ARES-RL throughout the pre-training phase. By incorporating synthetic exploit-response pairs that simulated OWASP-class vulnerabilities, the reinforcement agent was subjected to a broader range of success and failure scenarios. This equitable exposure enhanced convergence during initial training episodes and diminished policy oscillation.

Quantitatively, episodes incorporating simulated data resulted in an average episode reward increase of 14.3% compared to runs trained exclusively on actual test environments. The policy entropy of the RL agent declined consistently after 1.5 million timesteps, indicating that the feedback loop was converging towards optimal exploitative behavior.

These findings confirm that mixing real and synthetic samples within reinforcement-based cybersecurity frameworks is a viable and effective approach, particularly in scenarios where access to large, labeled vulnerability datasets is limited.

4.5 Discussion

The results confirm that combining a domain-adapted LLM, reinforcement learning control, and Hexstrike-AI orchestration within ARES meaningfully improves both reasoning quality and detection accuracy. Compared to conventional penetration testing tools, ARES brings a deeper level of contextual understanding to the table, reasoning about how payloads actually behave rather than simply matching against known signatures. The reinforcement loop allows the system to continuously optimize, learning over time which exploit strategies tend to work best, while the LLM provides clear, explainable reasoning behind each decision it makes.

In summary:

- The hybrid ARES-RL + LLM architecture improved model reliance and reduced false negatives.
- Synthetic simulation significantly contributed to policy stability.
- The framework demonstrated consistent accuracy (≈85%) and robustness across all test environments.

These outcomes collectively highlight ARES as a viable prototype for autonomous, ethical, and adaptive web penetration testing, bridging the gap between traditional vulnerability scanning and intelligent exploit reasoning.

5 Challenges

Building and assessing ARES presented a number of practical challenges that are worth discussing, as they influence the framework's current limitations and future enhancements.

The first challenge came from fine-tuning ARES v1 itself. Training a Llama 3 (8B) model on a 70 GB corpus of OWASP, CWE, and ExploitDB data required LoRA and 8-bit quantization just to keep GPU memory usage within reach. This helped make training possible on limited hardware, but also underlines that the fine-tuning process depends heavily on computational resources.

There was a genuine trade-off between false positives and false negatives when it comes to detection accuracy. Logistic Regression completely avoided false negatives while misclassifying 19 out of 49 benign samples. A significant proportion of ARES discoveries must still be manually verified in practical pentesting, which reduces the automation benefit the framework is supposed to offer.

Another source of conflict was integrating three distinct subsystems — Hexstrike-AI, the RL agent, and the LLM reasoner — without changing Hexstrike-AI's main source code. Maintaining five distinct agents (Recon, Analyzer, Exploit, RL, Policy) coordinated through a single controller introduces overhead and failure points inherent to using an asynchronous REST/WebSocket bridge.

6 Conclusion

This study introduced ARES (Automated Red-Teaming Evaluation System), a multi-agent penetration testing architecture that combines a domain-adapted LLM, a reinforcement learning agent, and the Hexstrike-AI automation engine.

The research showed that contextual reasoning and adaptive learning can meaningfully improve web application penetration testing over traditional static scanning methods. Findings demonstrated that ARES attained a consistent vulnerability detection accuracy of over ≈85%, exhibited low false-negative rates, and showed robust reinforcement convergence, while the integration of simulated data enhanced early policy stability by over 14%.

These findings substantiate the feasibility of integrating reasoning-driven models like Llama 3 (8B) with structured reinforcement contexts for ethical and autonomous online exploitation.

In addition to boosting accuracy, ARES offers a scalable design philosophy that efficiently integrates modern AI models into regular cybersecurity processes while maintaining ethics and operational standards. The proposed architecture remains modular and reproducible, allowing future researchers to extend the framework toward:

- automated threat intelligence feedback,
- cross-domain vulnerability analysis with greater large language models, and
- cooperative multi-agent learning enabling ongoing adaptation in real-time network settings.

ARES exemplifies a promising advancement in self-improving, interpretable, and policy-aware AI frameworks that can do secure and transparent penetration testing at scale.

Acknowledgements

Special appreciation is extended to the developers of the Hexstrike-AI open-source framework for making their platform accessible to the research community, and to contributors of publicly available cybersecurity datasets that enabled realistic and ethical experimentation. The authors also acknowledge the use of open-source machine learning tools including PyTorch, HuggingFace Transformers, and Gymnasium — that were instrumental in building and evaluating the ARES framework.

Declarations

- **Funding:** This research was not supported by any external funding agency. All tools, frameworks, and datasets used in this study — including Hexstrike-AI, PyTorch, HuggingFace Transformers, Gymnasium, Stable-Baselines3, and the OWASP, CWE, and ExploitDB datasets — are open-source and freely available, and the work was carried out without any commercial or proprietary licensing costs.
- **Competing interests:** The authors declare that there is no conflict of interest related to this work, either financial or non-financial.

- **Ethics approval and consent to participate:** All penetration testing activities described in this study were conducted exclusively against intentionally vulnerable, sandboxed applications (DVWA, OWASP Juice Shop, and bWAPP) running in isolated Docker containers on infrastructure controlled by the authors. No real-world systems, third-party infrastructure, or production environments were accessed or tested at any point. No human subjects, personal data, or personally identifiable information were involved in this research. All experiments complied with responsible and ethical AI usage guidelines, and the framework's built-in Policy Agent enforced whitelisted-target validation and rate-limited testing throughout evaluation.
- **Consent for publication:** Not applicable.
- **Data availability:** The datasets used for fine-tuning ARES v1 (OWASP, CWE, ExploitDB, and the Awesome Cybersecurity Datasets Repository, CSE-CIC-IDS2018, and CIC-IDS2017) are publicly available from their respective sources. The compiled fine-tuning corpus and trained model weights generated during this study are available from the corresponding author upon reasonable request.
- **Materials availability:** Not applicable.
- **Code availability:** The code developed in this study (ARES framework) is publicly available on GitHub at https://github.com/Sabby666/ARES.
- **Author contribution:** All authors contributed equally. T.M. conceived the research idea and led the development. All authors reviewed and approved the final manuscript.

References

[1] Bouafia, R., Benbrahim, H., Amine, A.: Automatic Protection of Web Applications Against SQL Injections: An Approach Based On Acunetix, Burp Suite and SQLMAP (2023). https://doi.org/10.1109/ICOA58279.2023.10308827

[2] 0x4m4: Hexstrike-AI – Offensive Security AI Framework (2024). https://github.com/0x4m4/hexstrike-ai

[3] Aruvasagachithan, A., Jeevanandh, D., Bharath, T., Nivas, A., Santhiya, R.M.: Automating web application penetration testing using ai. International Journal of Management, Engineering and Technology **2**(2) (2025) https://doi.org/10.5281/zenodo.15641579

[4] Bidve, V.S., Kakade, K., Paygude, P., Bidwe, R.: Evolution of automated penetration testing toolchains: Integration strategies and operational challenges. International Journal of Safety and Security Engineering (2025) https://doi.org/10.18280/ijsse.150720

[5] Al-Azzawi, M., Doan, D., Sipola, T., Hautamäki, J., Kokkonen, T.: Red teaming with artificial intelligence-driven cyberattacks: A scoping review (2025) https://doi.org/10.48550/arXiv.2503.19626 arXiv:2503.19626

[6] Yin, Z., Xu, Y., Ma, F., Gao, H., Qiao, L., Jiang, Y.: Scanner++: Enhanced vulnerability detection of web applications with attack intent synchronization.

ACM Transactions on Software Engineering and Methodology (2023) https://doi.org/10.1145/3517036

[7] Lupinacci, M., Pironti, F.A., Blefari, F., Romeo, F., Arena, L., Furfaro, A.: "the dark side of llms: Agent-based attacks for complete computer takeover" (2025) arXiv:2507.06850v5

[8] Shahrivar, P., Millar, S., Shereen, E.: Detecting web application dast attacks with machine learning. Rapid7 Research / KTH Royal Institute of Technology (2023) https://doi.org/10.1109/DSC61021.2023.10354106

[9] Kore, A., Indorkar, S., Hinduja, T., Wagh, S., Sawant, A., Rankhambe, S.: Burp suite extension for script-based attacks for web applications. In: 2022 International Conference on Emerging Trends in Engineering and Computer Applications (ICECA) (2022). https://doi.org/10.1109/ICECA55336.2022.10009116

[10] Mohan, D.K.R.: Comparative study of sast and dast tools with manual penetration testing methods for improving identification of business logic vulnerabilities. Master's thesis, National College of Ireland (2024). https://norma.ncirl.ie/id/eprint/8176

[11] Thaqi, R., Vishi, K., Rexha, B.: "enhancing burp suite with machine learning extension for vulnerability assessment of web applications". Journal of Applied Security Research (2022) https://doi.org/10.1080/19361610.2022.2096387

[12] Shahid, M.: Machine learning for detection and mitigation of web vulnerabilities and web attacks (2023) arXiv:2304.14451

[13] Yuan, X., Ding, L., Ben Salem, M., Li, X., Wu, D.: "connecting web event forecasting with anomaly detection: A case study on enterprise web applications using self-supervised neural networks" (2020) arXiv:2008.13707

[14] Diaz Parra, R.K.: Generative ai for web application pentesting. Issues in Information Systems **26**(1), 502–514 (2025) https://doi.org/10.48009/1_iis_2025_137

[15] Zhuravchak, A., Piskozub, A., Skorynovych, B., Lakh, Y.: "design and development of a large language model-based tool for vulnerability detection". Eastern European Journal of Enterprise Technologies **2**(2), 134 (2025) https://doi.org/10.15587/1729-4061.2025.325251

[16] Iturbe, E., Llorente-Vazquez, O., Rego, A., Rios, E., Toledo, N.: "unleashing offensive artificial intelligence: Automated attack technique code generation". Computers & Security **147**, 104077 (2024) https://doi.org/10.1016/j.cose.2024.104077

[17] Skandylas, C., Asplund, M.: Automated penetration testing: Formalization and

realization. Journal of Cybersecurity Research **14**(2), 45–58 (2025)

[18] López-Montero, D., Álvarez-Aldana, J.L., Morales-Martínez, A., Gil-López, M., Auñón-García, J.M.: Reinforcement learning for automated cybersecurity penetration testing. IEEE Access **13**, 12533–12550 (2025) https://doi.org/10.48550/arXiv.2507.02969

[19] Lupinacci, M., Pironti, F.A., Blefari, F., Romeo, F., Arena, L., Furfaro, A.: The dark side of llms: Agent-based attacks for complete computer takeover. arXiv preprint arXiv:2507.06850 (2025)

[20] Wang, L., Shi, X., Li, Z., Jiang, Y., Tan, S., Jiang, Y., Cheng, J., Chen, W., Shen, X., Li, Z., Chen, Y.: Automated penetration testing with llm agents and classical planning. arXiv preprint arXiv:2512.11143 (2025)

[21] Amouei, M., Rezvani, M., Fateh, M.: "rat: Reinforcement-learning-driven and adaptive testing for vulnerability discovery in web application firewalls". IEEE Transactions on Dependable and Secure Computing (2023) https://doi.org/10.1109/TDSC.2021.3095417

[22] Hu, Z., Beuran, R., Tan, Y.: "automated penetration testing using deep reinforcement learning". In: 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Genoa, Italy, pp. 42–51 (2020). https://doi.org/10.1109/EuroSPW51379.2020.00010

[23] Feffer, M., Sinha, A., Deng, W., Lipton, Z., Heidari, H.: "red-teaming for generative ai: Silver bullet or security theater?". In: AAAI Conference on Artificial Intelligence and Ethics, vol. 38, pp. 210–222 (2024). https://doi.org/10.48550/arXiv.2401.15897

[24] Harguess, J., Ward, C.M.: Offensive security for ai systems: Concepts, practices, and applications. IEEE Transactions on Information Forensics and Security **20**(5), 1350–1364 (2025)

[25] Sinha, A., Lucassen, J., Grimes, K., Feffer, M., Heidari, H.: What can generative ai red-teaming learn from cyber red-teaming? Technical report, Carnegie Mellon University, Software Engineering Institute (2025). https://www.sei.cmu.edu/documents/6301/What_Can_Generative_AI_Red-Teaming_Learn_from_Cyber_Red-Teaming.pdf

[26] Familoni, B.T.: "cybersecurity challenges in the age of ai: Theoretical approaches and practical solutions". Computer Science & IT Research Journal **5**(3), 703–724 (2024) https://doi.org/10.51594/csitrj.v5i3.930

[27] Alauthman, M., Almomani, A., Aoudi, S., Al-Qerem, A., Aldweesh, A.: "Automated Vulnerability Discovery: Generative AI in Offensive Security", pp. 122–142. IGI Global, ??? (2025). https://doi.org/10.4018/979-8-3373-0832-6.ch013

[28] Orpak, K.: "generative ai and cybersecurity: Exploring opportunities and threats at their intersection". Maandblad voor Accountancy en Bedrijfseconomie **99**(4), 211–229 (2025) https://doi.org/10.5117/mab.99.149299

[29] Valencia, L.J.: "artificial intelligence as the new hacker: Developing agents for offensive security". Master's thesis, New Mexico Institute of Mining and Technology (2024). https://doi.org/10.48550/arXiv.2406.07561 . https://doi.org/10.48550/arXiv.2406.07561

[30] Antar, S.S.: Pioneering autonomous penetration testing with large language models through prompt engineering and agentic system design. PhD thesis, Queen's University (2025). https://qspace.library.queensu.ca/bitstreams/15e89303-70b1-427b-9fb0-fb02984fae24/download

[31] Girhepuje, S., Verma, A., Raina, G.: A survey on offensive ai within cybersecurity (2024) arXiv:2410.03566

[32] Al-Sinani, H.S., Mitchell, C.J.: "pentest++: Elevating ethical hacking with ai and automation" (2025) arXiv:2502.09484

[33] David, I., Gervais, A.: Multi-agent penetration testing ai for the web (2025) https://doi.org/10.48550/arXiv.2508.20816 arXiv:2508.20816

[34] Alquwayzani, A., Aldossri, R., Frikha, M.: Mitigating security risks in firewalls and web applications using vulnerability assessment and penetration testing (vapt). International Journal of Advanced Computer Science and Applications **15**(5) (2024) https://doi.org/10.14569/IJACSA.2024.01505136

[35] Choudhary, R., Rawat, J., Singh, G.: "comprehensive exploration of web application security testing with burp suite tools". International Journal for Multidisciplinary Research **5**(6) (2023) https://doi.org/10.36948/ijfmr.2023.v05i06.11297

[36] Sánchez-Collado, G., Olayinka, O., Pasikhani, A.: Web application penetration testing with artificial intelligence: A systematic review. In: National Conference on Automation (NCA 2024) (2024). https://doi.org/10.1109/NCA61908.2024.00043

[37] Hilario, E., Azam, S., Sundaram, J., Mohammed, K.I., Shanmugam, B.: "generative ai for pentesting: The good, the bad, the ugly". International Journal of Information Security **23**(3), 2075–2097 (2024) https://doi.org/10.1007/s10207-024-00835-x

[38] Mathavan, A.S.: Ai-augmented red teaming: Leveraging evolutionary algorithms in penetration testing methodologies. Journal of Artificial Intelligence and Cyber Security (JAICS) **9**(1) (2025)

[39] Anis, F., Hammoudeh, M.: Weaponizing ai in cyberattacks: A comparative study of ai-powered tools for offensive security. In: The 8th International Conference on

Future Networks & Distributed Systems (ICFNDS '24), Marrakech, Morocco, pp. 1–8 (2024). https://doi.org/10.1145/3726122.3726164

[40] Happe, A., Cito, J.: On the surprising efficacy of llms for penetration-testing (2025) arXiv:2507.00829

[41] Thapaliya, S., Dhital, S.: Ai-augmented penetration testing: A new frontier in ethical hacking. International Journal of Atharva **12**(1), 55–68 (2025)

[42] Ubagaram, C., Dyavani, N.R., Jayaprakasam, B.S., Mandala, R.R., Garikipati, V., Kumar, V.: Ethical Hacking and Penetration Testing: Strengthening Cyber Defense with AI-Driven Breach and Attack Simulation (2023). https://www.academia.edu/download/123219018/Ethical_Hacking_and_Penetration_Testing_Strengthening_Cyber_Defense_with_AI_Driven_Breach_and_Attack_Simulation.pdf

[43] Bhagwat, D., Saha, S., Subramanian, M., et al.: Pyrit: A framework for security risk identification and red teaming in generative ai systems (2024) https://doi.org/10.48550/arXiv.2410.02828 arXiv:2402.10357

[44] Kumar, A., Nanda, S., Singh, R.: "review into state of the art of vulnerability assessment using artificial intelligence". In: Artificial Intelligence and Evolutionary Computations in Engineering Systems, pp. 451–461. Springer, Singapore (2018). https://doi.org/10.1007/978-3-319-92624-7_1

[45] Deptula, K.: Automation of cyber penetration testing using the detect, identify, predict, react intelligence automation model. Master's thesis, Naval Postgraduate School, Monterey, CA, USA (2013). https://hdl.handle.net/10945/37612

[46] Mulla, R., Vincent, A.D., Greunke, B., Landers, N., Palm, B., Pearce, W.: The automation advantage in ai red teaming (2025) arXiv:2504.19855v2

[47] Majumdar, S., Pendleton, B., Gupta, A.: Red teaming ai red teaming. In: Proceedings of Machine Learning Research, vol. 299, pp. 1–20 (2025). https://arxiv.org/abs/2507.05538

[48] Al-Azzawi, M.: Artificial Intelligence in Red Teaming. https://urn.fi/URN:NBN:fi:amk-2024122037866

[49] Phanireddy, S.: Securing modern web applications using ai-driven static and dynamic analysis techniques. International Journal of Artificial Intelligence, Data Science, and Machine Learning **6**(2), 73–82 (2025) https://doi.org/10.63282/3050-9262.IJAIDSML-V6I2P108

[50] Wen, S.-F., Shukla, A., Katt, B.: "artificial intelligence for system security assurance: A systematic literature review". International Journal of Information Security **24**, 43–82 (2025) https://doi.org/10.1007/s10207-024-00959-0

[51] Sahil, S.G.K., Chavan, C.P.: Automated penetration testing tool. In: 2025 World Skills Conference on Universal Data Analytics and Sciences (WorldSUAS), Bangalore, India, pp. 1–8 (2025). https://ieeexplore.ieee.org/document/11199262

[52] Khan, S.A., Azim, N., Iqbal, A., Abbas, H., Qureshi, S.: Securing web applications: A practical approach to mitigating owasp top 10 vulnerabilities. VAWKUM Transactions on Computer Sciences **11**(2) (2025) https://doi.org/10.21015/vtse.v13i2.2145