JOURNAL OF THE ROYAL LAUREATES ACADEMY
www.rlaindia.org

EXPERIMENTAL EVALUATION OF AGENTGUARD AI AGAINST BASELINE RUNTIME CONFIGURATIONS UNDER CONTROLLED LLM-AGENT WORKLOADS

Deepak Kumar Gour
Department of Computer Science and Engineering, Jayoti Vidyapeeth Women's University, Jaipur, Rajasthan, India

Dr. Sushma Agrawal
Research Supervisor, Jyoti Vidyapeeth Woman's University, Jaipur, Rajasthan, India

ABSTRACT

This paper experimentally evaluates AgentGuard AI, an integrated runtime-security framework combining the Secure Runtime Monitoring Protocol (SRMP), Adaptive Agent Risk Scoring (AARS), and Risk-Adaptive Scheduling and State Control (RASS). The evaluation uses the preserved AgentGuard AI v2.4.0 experiment suite and compares the full pipeline against reduced runtime configurations under controlled synthetic LLM-agent workloads. Seven primary runs of 200 events examine attack-rate conditions from 0% to 50%, a matched equal-weight AARS ablation, same-seed replay, and alternate-seed sensitivity; a ten-seed extended suite adds 2,000 further event evaluations. In the primary 30% condition, the integrated evaluator reports precision 0.9048, recall 0.7917, F1-score 0.8444, and AUC-ROC 0.9996. The full configuration exceeds the meaningful no-monitoring and threshold-only baselines, while replacing the canonical AARS weights with uniform weights reduces F1 to 0.3636. RASS block recommendations increase from 0.5% in the zero-attack condition to 32.5% at 50% configured attack prevalence, showing proportional response within the simulator. Across ten fixed seeds, mean F1 is 0.791 with a reported 95% interval of [0.760, 0.819]. The results support the internal reproducibility and comparative value of the integrated design while also exposing important evidence boundaries: the workloads are synthetic, several legacy baselines are artefactually strong, and the latency

field represents simulated workload behaviour rather than measured middleware overhead. No independently verified real-workload trace is available in the preserved evidence package, so real-world performance is not claimed.

Keywords—AgentGuard AI; LLM agents; runtime security; baseline comparison; experimental evaluation; AARS; RASS; resource-constrained computing.

I. INTRODUCTION

Large language model applications increasingly operate as agents that retrieve external information, call tools, maintain memory, and execute multi-step actions. This shift creates security risks that emerge during execution rather than only at model-training time. Prompt injection, indirect context manipulation, jailbreaks, unsafe tool use, and excessive agency have therefore become central concerns in LLM application security [1]-[8].

AgentGuard AI was developed as a model-agnostic research framework for this runtime layer. SRMP produces transparent security evidence, AARS converts heterogeneous event features into a six-dimensional composite risk score, and RASS maps the assessed risk to proportionate operational states and actions. Earlier framework and algorithmic papers can examine these components individually; the present paper asks a different question: does the integrated pipeline behave more usefully than reduced or unprotected configurations under controlled workloads?

The contribution of this paper is an evidence-based comparative evaluation using the actual experiment outputs preserved with the thesis. No new values are assumed or fabricated. The paper focuses on four aspects: performance across attack-rate conditions, comparison with meaningful baseline configurations, stability across fixed seeds, and proportionality of the RASS response. A final section distinguishes what the current application evidence can support from what remains untested.

II. EXPERIMENTAL DESIGN

A. System under Test

The evaluated prototype is AgentGuard AI v2.4.0 running the fixed SRMP-RuleLib-1.0.0, AARS-1.0.0, and RASS-SM-1.0.0 configurations. SRMP contains 23 rules across eight threat categories. AARS uses six weighted dimensions, while RASS provides risk-adaptive

operational recommendations. The system was executed in a CPU-only Replit research environment without an external risk-scoring API.

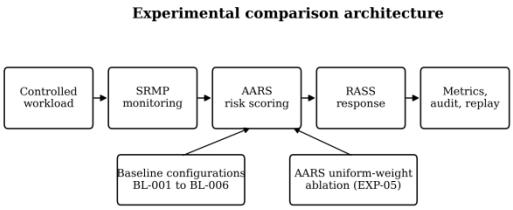

Fig. 1. Experimental comparison architecture for the AgentGuard AI evaluation.

B. Primary and Extended Protocol

The preserved primary protocol contains seven runs, each with 200 synthetic events. EXP-01 to EXP-04 vary the configured attack prevalence while holding seed 42 and the canonical configuration fixed. EXP-05 repeats the 30% condition with uniform AARS weights. EXP-06 is a same-seed replay of the primary condition, and EXP-07 repeats the 30% condition with seed 123. The extended suite contains ten additional fixed-seed runs at 30% attack prevalence, adding 2,000 event evaluations.

TABLE I. Experiment Matrix

Run	Setup	Purpose
EXP-01	0%, seed 42, canonical	Zero-injection control
EXP-02	10%, seed 42, canonical	Low prevalence
EXP-03	30%, seed 42, canonical	Primary condition
EXP-04	50%, seed 42, canonical	High prevalence
EXP-05	30%, seed 42, uniform AARS	Weight ablation
EXP-06	30%, seed 42, canonical	Same-seed replay
EXP-07	30%, seed 123, canonical	Alternate seed
MEXP-01..10	30%, ten fixed seeds	Cross-seed stability

C. Baseline Definitions

The thesis package defines six runtime comparison configurations. BL-001 disables SRMP, AARS, and RASS and therefore acts as the unprotected reference. BL-003 is a threshold-only reduced system without the full security dimension and is the most useful reduced baseline. BL-006 is the complete AgentGuard pipeline. EXP-05 provides a matched AARS ablation in which the canonical weights are replaced by equal weights.

Three additional baselines require caution. BL-002 static signature monitoring, BL-004 security-only scoring, and BL-005 static fixed response all achieve perfect scores in the closed simulator. The preserved analysis identifies these results as lexical or operational artefacts rather than credible evidence of superiority. Consequently, direct comparative claims in this paper rely primarily on BL-001, BL-003, EXP-05, and BL-006.

D. Evaluation Measures and Evidence Boundary

The preserved evaluator reports confusion-matrix metrics, AUC-ROC, risk-score distributions, RASS action counts, run-level uncertainty summaries, and reproducibility checks. Construct-validity issue remains: the statistical package uses a security-flag-derived reference in the main confusion matrices, whereas the simulator also contains an independently generated is_attack field. These representations are not equivalent. The classification results are therefore described as evaluator-level decision alignment rather than independent estimates of attack-truth accuracy.

A second boundary concerns runtime performance. The latency_ms field is generated as part of the synthetic workload and is used in the infrastructure-risk dimension. It is not a direct wall-clock measurement of SRMP-AARS-RASS middleware overhead. CPU and memory values are retained as research environment observations, but the present paper does not claim measured production throughput or real-user workload performance.

III. RESULTS

A. Behaviour Across Attack-Rate Conditions

The integrated pipeline produces stronger evaluator-level decision metrics as the attack prevalence rises from the low-prevalence condition to the primary and high-prevalence conditions. EXP-02 reports precision 0.5909, recall 0.8667, and F1 0.7027. Under EXP-03, precision rises to 0.9048, recall is 0.7917, and F1 reaches 0.8444. EXP-04 produces precision 0.9123, recall 0.8000, and F1 0.8525. The zero-attack control contains fourteen High/Critical decisions among 200 events, corresponding to an evaluator-level false-positive rate of 7.0%.

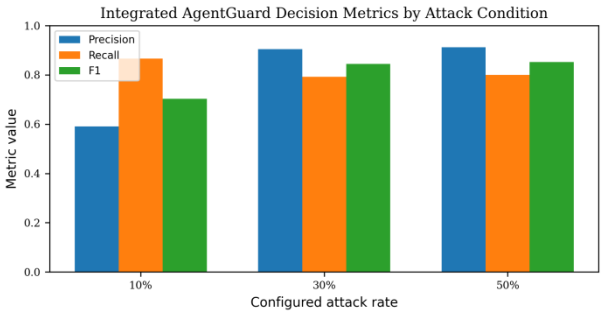

Fig. 2. Precision, recall, and F1 across the 10%, 30%, and 50% attack-rate conditions.

B. Baseline and Ablation Comparison

The full pipeline shows a clear advantage over the meaningful reduced configurations in the preserved experiment. BL-001 provides no threat classification and has F1 0.000. BL-

003, which removes the full security contribution and relies on a reduced threshold-only design, records F1 0.333 and AUC-ROC 0.906. The equal-weight AARS ablation records F1 0.3636 and recall 0.2500. By comparison, BL-006 / EXP-03 reports F1 0.8444 and AUC-ROC 0.9996.

The ablation is particularly informative because the AUC of the uniform-weight system remains 0.9966. This means the reduced configuration still ranks many events correctly, but its fixed threshold converts those scores into substantially poorer operational decisions. The comparison therefore supports the canonical weighting structure without requiring claims based on the artefactually perfect static-signature or security-only baselines.

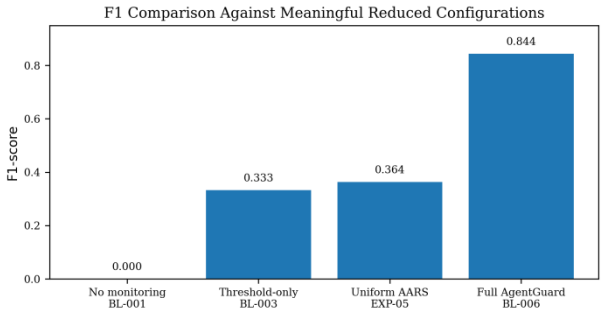

Fig. 3. F1 comparison between the full AgentGuard configuration and meaningful reduced baselines.

TABLE II. Comparative Baseline Evidence

Configuration	F1 / AUC-ROC	Interpretation
BL-001 No monitoring	0.000 / 0.500	Unprotected reference
BL-003 Threshold-only	0.333 / 0.906	Meaningful reduced baseline
EXP-05 Uniform AARS	0.364 / 0.9966	Matched weight ablation
BL-006 Full AgentGuard	0.844 / 0.9996	Integrated framework

C. Adaptive Response Behaviour

RASS becomes more restrictive as the configured attack rate increases. The number of block recommendations is 1, 15, 48, and 65 for the 0%, 10%, 30%, and 50% conditions respectively. In rate terms, blocking rises from 0.5% to 32.5%. This monotonic pattern supports the intended proportionality of the control layer inside the simulator.

The result should not be interpreted as proof that the selected actions are optimal. Some restriction also occurs in the zero-attack condition: EXP-01 includes 109 sandbox recommendations, seven throttles, and one block. These outputs illustrate why adaptive-

security evaluation must consider the cost of unnecessary restriction alongside threat detection.

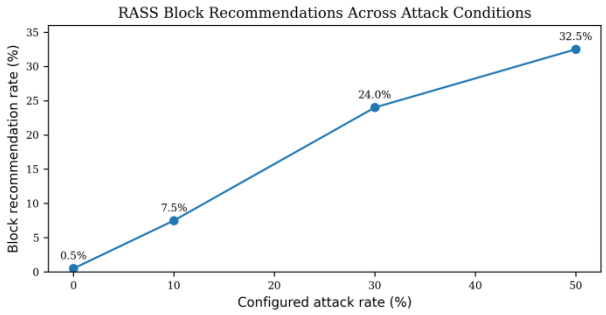

Fig. 4. RASS block recommendation rate as configured attack prevalence increases.

D. Reproducibility and Cross-Seed Stability

EXP-06 repeats the EXP-03 seed and configuration and reproduces precision, recall, F1, AUC-ROC, and mean risk to the reported precision. This confirms deterministic execution under the preserved configuration. EXP-07 changes the seed to 123 and produces F1 0.8372, only 0.0072 below the primary run, while mean risk changes by 0.0108.

The ten-seed suite provides a broader stability view. Mean F1 is 0.791 with a reported 95% percentile-bootstrap interval of [0.760, 0.819]; mean precision is 0.874, mean recall 0.725, mean AUC-ROC 0.9999, and mean Cohen's d 4.636. F1 ranges from 0.674 to 0.844 across the fixed seeds, indicating meaningful run-to-run variation despite near-ceiling rank separation.

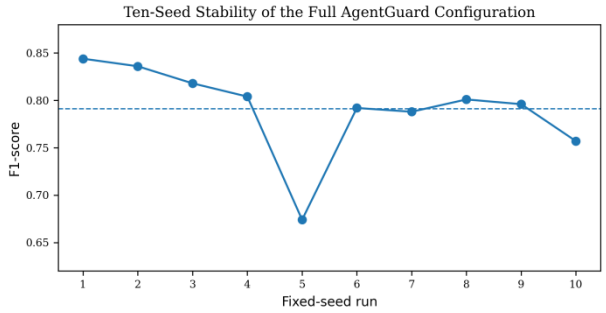

Fig. 5. F1-score across the ten fixed-seed full-pipeline runs; dashed line indicates the mean (0.791).

E. Resource and Timing Interpretation

Mean CPU utilisation across the principal attack-rate runs stays near 51-52%, while mean memory utilisation remains between 54% and 57%. No external risk-scoring service is

invoked. These observations support the tested CPU-oriented execution requirement at the prototype level.

TABLE III. Tested Environment Profile

Run	CPU / memory	Latency / P95
EXP-01	52.2% / 54.1%	669.7 / 1151.9 ms
EXP-02	52.5% / 54.7%	806.5 / 2466.4 ms
EXP-03	51.0% / 54.0%	1163.8 / 2836.4 ms
EXP-04	52.3% / 57.0%	1355.5 / 3194.4 ms

The latency values in Table III must not be interpreted as measured security-middleware overhead. They are simulation-generated workload features, and attack events receive additional synthetic latency. A future real-workload study must separately time SRMP, AARS, RASS, and end-to-end application execution using a high-resolution wall-clock measurement.

F. Supplementary Robustness and Transfer Checks

The primary experiment is intentionally closed and reproducible, so supplementary analyses were used to test whether the observed behaviour survives outside the exact lexical conditions of the simulator. In the evasion study, the original attack prompts are detected at 76%. Case variation and benign prefix/suffix noise do not reduce this rate, which is consistent with the case-insensitive matching design. Synonym substitution reduces detection to 12%, while leet-speak, whitespace manipulation, and Unicode homoglyphs reduce detection to 16%. A combined synonym-plus-leet transformation again produces 12% detection. These results show that lexical robustness, rather than aggregate simulator performance, is one of the principal constraints on transfer.

A separate 64-prompt HarmBench-inspired research set provides a small out-of-vocabulary check. It contains 34 harmful and 30 benign prompts and is not the official full HarmBench benchmark. On this set, SRMP reports precision 1.000, recall 0.5294, and F1 0.6923. The combination of perfect precision and limited recall is consistent with a conservative rule library that avoids many false positives but misses differently phrased malicious intent. This result reinforces the need to interpret the near-ceiling AUC values of the simulator together with transfer and evasion evidence.

These supplementary findings also clarify the proper role of the baseline experiment. The controlled protocol is valuable for isolating the effect of AARS weighting, RASS policy, seed choice, and attack prevalence, but it cannot by itself establish semantic robustness. A credible real-workload extension should therefore preserve the reproducibility controls of the current

experiment while introducing independent labels, open-vocabulary traffic, adaptive attacks, and measured per-stage runtime overhead.

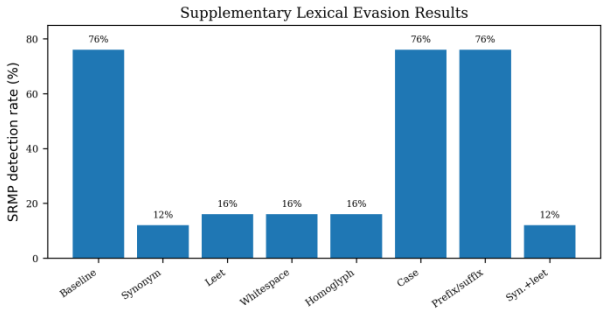

Fig. 6. Detection rate under the preserved supplementary lexical-evasion transformations.

IV. COMPARATIVE CONTEXT WITH PUBLISHED AGENT-SECURITY EVALUATIONS

The AgentGuard experiment is not directly comparable with modern agent benchmarks because the evaluation units differ. AgentDojo evaluates realistic user tasks in tool-enabled environments and explicitly studies the security-utility trade-off [6]. InjecAgent focuses on indirect prompt injection across 17 user tools and 62 attacker tools [7], while Agent Security Bench broadens the threat surface to system prompts, user prompts, tools, memory, and mixed attack stages [8]. Their reported attack-success or utility measures should therefore not be placed in the same numerical table as the evaluator-level F1 values produced by AgentGuard.

Even so, these benchmarks are useful for interpreting the present results. First, they confirm that security evaluation must extend beyond one prompt classifier: tool authority, retrieval context, memory, and downstream actions materially change the attack surface. AgentGuard addresses this systems perspective by combining SRMP evidence, six-dimensional AARS scoring, and RASS control, but the current simulator does not reproduce the full environments of AgentDojo, InjecAgent, or ASB.

Second, the benchmark literature reinforces the need to measure utility alongside security. The RASS experiment demonstrates monotonic escalation, but the high number of sandbox recommendations in the zero-attack condition shows that security actions can impose operational cost even when no attack is injected. A future comparison should therefore report

benign-task completion, false restriction, review burden, and intervention latency in addition to detection metrics.

Third, structured defences such as StruQ and CaMeL illustrate that prevention and runtime monitoring solve different parts of the problem [11], [12]. StruQ strengthens the boundary between trusted instructions and untrusted data, while CaMeL constrains control and data flow through capabilities. AgentGuard instead observes runtime evidence and adapts response. A realistic future study should evaluate these layers together rather than treating them as mutually exclusive alternatives.

The current paper therefore positions AgentGuard as an experimentally reproducible runtime-control architecture, not as a replacement for agent benchmarks or secure-by-design application controls. Its comparative claim is limited to the preserved baseline configurations executed under the same controlled simulator. Broader superiority claims would require common workloads, common models, common metrics, and independently reproduced implementations.

V. DISCUSSION

The strongest finding is not a single high metric but the comparative behaviour of the integrated design. Full AgentGuard materially exceeds the unprotected and threshold-only configurations, and the matched AARS ablation shows that the canonical weighting scheme affects thresholded decisions. At the same time, the perfect results produced by several lexical baselines demonstrate why closed simulation vocabularies can create misleading comparisons.

The RASS results show that the architecture does more than classify events: it converts changing risk density into different operational responses. This is important because a runtime security system must ultimately decide how an agent should proceed. However, the zero-attack restrictions also show that proportionality alone is not sufficient; operational validation must quantify unnecessary restriction, human-review burden, and task success.

The cross-seed results support reproducibility within one implementation and simulator, but they do not establish independent reproducibility across institutions, datasets, or model providers. The same caution applies to the very high AUC values, which are partly a consequence of strong separation within the synthetic data-generating process. The present

evidence therefore supports an experimentally reproducible research prototype rather than production-certified security.

VI. LIMITATIONS AND REAL-WORKLOAD READINESS

The absence of real workload traces is treated here as a methodological boundary rather than a missing number to be estimated. The verified application evidence supports controlled simulation, replay, ablation, and internal robustness analysis. It does not contain a source-identified enterprise traffic corpus, independent human labels, or measured live LLM/tool execution timings. Accordingly, all discussion of real-workload validation in this paper is prospective.

The preserved evidence package does not contain an independently verified corpus of real organisational or user workload traces. For that reason, this paper deliberately does not present a real-vs-simulated numerical comparison, even though such a comparison is an important next research step. Introducing invented 'real' values would violate the evidentiary boundary of the thesis and would make the experimental claim indefensible.

A proper real-workload extension should use independently labelled traffic, preserve data provenance, separate calibration from test data, measure true wall-clock overhead, and retain both security and utility outcomes. At minimum, the full pipeline should be compared with an unprotected runtime, a lightweight classical detector, and any selected semantic guard under the same workload and hardware conditions. Until those data exist, the current paper remains a controlled comparative experiment.

TABLE IV. Main Validity Limitations

Validity issue	Evidence boundary / required next control
Synthetic English workloads	Open-world transfer unknown; use independent labelled workloads.
Evaluator-label ambiguity	Decision alignment is not attack truth; audit labels independently.
Closed-vocabulary baselines	Simple detectors may be inflated; use out-of-vocabulary testing.
Simulated latency	No measured middleware overhead; time each stage directly.
Single environment	External reproducibility untested; require independent replication.

VII. CONCLUSION

This paper evaluated AgentGuard AI as an integrated SRMP-AARS-RASS runtime-security pipeline against the baseline and ablation configurations preserved in the research application. Under the primary 30% condition, the integrated evaluator reports F1 0.8444 and AUC-ROC 0.9996. The full system exceeds the no-monitoring, threshold-only, and uniform-weight configurations, while RASS becomes progressively more restrictive as configured attack prevalence increases.

The ten-seed suite and same-seed replay support reproducibility within the simulator, but the study also identifies clear limitations: closed-vocabulary effects, evaluator-label ambiguity, simulated latency, and the absence of verified real workload traces. The evidence therefore supports AgentGuard AI as a reproducible experimental framework for runtime risk control, not as proof of production efficacy. The next experimental stage should introduce independent real workloads and direct per-stage runtime measurement before stronger deployment claims are considered.

REFERENCES

[1] A. Vaswani et al., "Attention Is All You Need," in Advances in Neural Information Processing Systems, vol. 30, 2017.

[2] T. B. Brown et al., "Language Models are Few-Shot Learners," in Advances in Neural Information Processing Systems, vol. 33, pp. 1877-1901, 2020.

[3] S. Yao et al., "ReAct: Synergizing Reasoning and Acting in Language Models," in Proc. Int. Conf. Learn. Representations (ICLR), 2023.

[4] F. Perez and I. Ribeiro, "Ignore Previous Prompt: Attack Techniques for Language Models," arXiv:2211.09527, 2022.

[5] K. Greshake et al., "Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection," arXiv:2302.12173, 2023.

[6] E. Debenedetti et al., "AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents," in Advances in Neural Information Processing Systems, vol. 37, 2024.

[7] Q. Zhan et al., "InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents," in Findings of ACL 2024, pp. 10471-10506, 2024.

[8] H. Zhang et al., "Agent Security Bench (ASB): Formalizing and Benchmarking Attacks and Defenses in LLM-Based Agents," in Proc. Int. Conf. Learn. Representations (ICLR), 2025.

[9] H. Inan et al., "Llama Guard: LLM-Based Input-Output Safeguard for Human-AI Conversations," arXiv:2312.06674, 2023.

[10] T. Rebedea et al., "NeMo Guardrails: A Toolkit for Controllable and Safe LLM Applications with Programmable Rails," arXiv:2310.10501, 2023.

[11] S. Chen et al., "StruQ: Defending Against Prompt Injection with Structured Queries," in Proc. 34th USENIX Security Symp., pp. 2383-2400, 2025.

[12] E. Debenedetti et al., “Defeating Prompt Injections by Design,” arXiv:2503.18813, 2025.

[13] National Institute of Standards and Technology, Artificial Intelligence Risk Management Framework (AI RMF 1.0), NIST AI 100-1, 2023.

[14] National Institute of Standards and Technology, Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile, NIST AI 600-1, 2024.

[15] OWASP Foundation, OWASP Top 10 for LLM Applications 2025, OWASP Generative AI Security Project, Version 2.0, 2025.

[16] MITRE Corporation, “MITRE ATLAS: Adversarial Threat Landscape for Artificial Intelligence Systems,” 2024.