Toward Dynamic and Risk-Aware Evaluation of Cybersecurity LLMs: A Survey and the RIRAG Framework

Ravi Prasad[1, *], Feroz Ahmed[2], Sujan Kumar Reddy Challa[4], Aditya Garg[5], Md Shohel Rana[3], and Charan Gudla[1]

[1]*Department of Computer Science and Engineering, Mississippi State University, Mississippi State, MS 39762, USA*

[2]*Prediction 3D Technologies, Biloxi, MS, USA*

[3]*School of Computing, Georgia Southern University, Statesboro, GA 30460, USA*

[4]*Independent Researcher, USA*

[5]*Independent Researcher, USA*

Abstract: The rapid adoption of large language models (LLMs) in cybersecurity has created a growing need for evaluation methods that reflect operational risk rather than isolated language capability. Existing cybersecurity benchmarks assess useful dimensions such as factual knowledge, vulnerability analysis, secure coding, penetration testing, and threat intelligence reasoning, but many remain limited by static datasets, weak diagnostic granularity, limited adversarial testing, and insufficient attention to human-AI decision-making. This survey analyzes recent LLM cybersecurity benchmarks through three evaluation paradigms: knowledge-oriented, task-oriented, and holistic evaluation. From this analysis, we identify five recurring gaps between benchmark performance and real-world cybersecurity risk: knowledge-reasoning mismatch, capability-risk separation, limited failure attribution, staticity and contamination, and adversarial fragility. To address these gaps, we introduce the Reflective and Iterative Retrieval-Augmented Generation (RIRAG) framework, a dynamic and risk-aware evaluation architecture for cybersecurity LLMs. RIRAG combines continuously updated cybersecurity knowledge, retrieval- and generation-specific metrics, diagnostic logging, independent evaluation, adversarial red teaming, operational risk scoring, and human-AI teaming assessment. The framework is specified through design principles, formal components, implementation guidance, and illustrative case studies. Rather than presenting RIRAG as a validated production system, this article defines a falsifiable empirical validation protocol for future study. The central contribution is a survey-grounded reference architecture for evaluating cybersecurity LLMs as evolving, adversarially exposed, and human-interactive systems.

Keywords: Large Language Models, Cybersecurity Benchmarking, Retrieval-Augmented Generation, Dynamic Benchmarking, Adversarial Robustness, Operational Risk Assessment, Human-AI Teaming, Cyber Threat Intelligence.

INTRODUCTION

Large Language Models (LLMs) are rapidly becoming integral to modern cybersecurity, supporting Security Operations Center (SOC) operations, cyber threat intelligence, vulnerability assessment, secure software development, and incident response [1]. Commercial platforms such as Microsoft's Copilot for Security and Google's Gemini in Security further illustrate their growing role as decision-support systems that augment human analysts [2–5].

The same capabilities also create new security risks. LLMs are vulnerable to prompt injection, jailbreak attacks, and training- or retrieval-data poisoning, while simultaneously enabling offensive tasks such as reconnaissance, phishing, malware development, and exploit generation [6–13]. Recent studies further demonstrate that coordinated LLM agents can identify and exploit both one-day and previously unknown vulnerabilities with minimal human intervention [14–16].

This dual-use nature makes rigorous evaluation essential. Strong performance on general language benchmarks does not necessarily translate into reliable behavior in operational cybersecurity environments, where decisions are adversarial, time-sensitive, and safety-critical. Evaluation must therefore extend beyond language capability to measure robustness, operational effectiveness, and human-AI collaboration.

Why Specialized Evaluation Matters

The rapid adoption of LLMs has been accompanied by an equally rapid expansion of evaluation benchmarks. General-purpose benchmarks such as GLUE, MMLU, and HELM remain valuable for measuring language understanding and reasoning [17–19], but they provide only limited insight into cybersecurity tasks, which require procedural reasoning, code understanding, multi-stage attack

*Address correspondence to this author at the Department of Computer Science and Engineering, Mississippi State University, Mississippi State, MS 39762, USA;
E-mail: rp1416@msstate.edu

analysis, and decision making under adversarial conditions [10,20].

To address this gap, researchers have developed cybersecurity-specific benchmarks covering vulnerability detection, malware analysis, penetration testing, cyber threat intelligence, secure code generation, SOC workflows, and agentic cyber operations [1,5,10,21]. These benchmarks provide a richer assessment of model behaviour but remain limited. Many emphasize factual knowledge over procedural reasoning, rely on static datasets that quickly become outdated, or evaluate models in isolation despite real deployments increasingly combining retrieval systems, external tools, autonomous agents, and human analysts.

Another challenge is that benchmark development has largely followed, rather than guided, operational deployment. As a result, benchmark scores often provide only limited evidence of operational robustness, adversarial resilience, or deployment risk [2,4,9,20,22]. Future cybersecurity evaluation should therefore assess LLMs as components of operational systems, incorporating realistic workflows, evolving knowledge sources, adversarial behaviour, human–AI collaboration, and the resilience of the evaluation process itself.

SCOPE AND THESIS

The literature on LLM cybersecurity benchmarking has expanded rapidly, yet existing surveys primarily focus on individual benchmark collections or specific application domains. A comprehensive synthesis that connects benchmark design with operational cybersecurity requirements remains limited.

This survey addresses that gap by organizing the literature into three complementary paradigms: knowledge-oriented, task-oriented, and holistic benchmarks. We analyze their assumptions, methodologies, strengths, and limitations, identify five structural gaps between benchmark performance and operational cybersecurity risk, and introduce the *Reflective and Iterative Retrieval-Augmented Generation* (RIRAG) framework as a methodology for more adaptive, risk-aware cybersecurity evaluation.

The contribution of this article is deliberately analytical and architectural rather than experimental. It is a survey combined with a reference architecture: a criterion-based taxonomy of the field, a derivation of design requirements from documented failure modes, a formally specified evaluation framework grounded in those requirements, and a falsifiable validation protocol (Section 9) that states the hypotheses, datasets, baselines, and analyses by which the framework's claims can be tested. Prototype implementation and the experimental execution of that protocol are explicitly scoped as future work; the framework should therefore be read as a specification against which empirical results can later be judged, not as a validated production system.

CONTRIBUTIONS

This paper makes the following contributions:

1. **Presents a criterion-based taxonomy** of LLM cybersecurity benchmarks organized into knowledge-oriented, task-oriented, and holistic evaluation paradigms, with explicit inclusion criteria, a defined classification dimension, and rules for resolving boundary cases (Section 3.1).
2. **Develops a threat model** for LLM-assisted cybersecurity systems, identifies five structural gaps separating current benchmark performance from operational cybersecurity risk, and derives from each gap an explicit design requirement together with the alternative designs considered and rejected (Section 4.5).
3. **Introduces the RIRAG framework**, including the evaluation-theoretic foundations of its seven design principles (Section 5.1), its formal specification, algorithms, evaluation metrics, and four extensions covering adversarial red teaming, operational risk assessment, human–AI teaming evaluation, and knowledge-graph-enhanced retrieval.
4. **Provides an implementation blueprint** and four end-to-end case studies demonstrating the proposed evaluation methodology.
5. **Defines a falsifiable validation protocol and research agenda**, stating the hypotheses, datasets, baselines, and validity controls for future empirical study of the framework, together with deployment considerations and ethical challenges.

Paper Organization

The remainder of this article is organized as follows. Section 2 reviews the technical foundations, threat model, and survey methodology. Section 3 presents the benchmark taxonomy together with its construction criteria. Section 4 analyzes current evaluation gaps and derives the design requirements that follow from them. Section 5 introduces the RIRAG framework - its

theoretical foundations, design principles, notation, and the coupled Knowledge and Evaluation Circuits. Section 6 presents the framework's four extensions: adversarial red teaming, operational risk modeling, human–AI teaming, and graph-enhanced retrieval with active learning. Section 7 describes the implementation blueprint, case studies, and failure diagnostics. Section 8 discusses deployment scenarios and the attack–defense landscape, and Section 9 defines the empirical validation protocol. Section 10 positions RIRAG within related work, Section 11 discusses limitations, ethics, and future directions, and Section 12 concludes.

Foundations

LLM cybersecurity evaluation is shaped by three closely related technologies: the language models themselves, retrieval-augmented generation (RAG), and agentic systems. Each introduces distinct capabilities, failure modes, and attack surfaces that influence how cybersecurity benchmarks should be designed. This section reviews these foundations, presents a threat model for LLM-assisted cybersecurity systems, and outlines the methodology used to identify and analyze the surveyed benchmarks.

Large Language Models

Large Language Models (LLMs) have become the foundation of AI-assisted cybersecurity because they support a broad range of language-intensive security tasks without task-specific model redesign. Their evaluation differs from that of conventional machine learning systems because performance depends heavily on previously observed data rather than explicit rules.

Two characteristics are particularly important for cybersecurity benchmarking. First, LLM competence is largely *distributional and interpolative*: models perform well on tasks resembling patterns seen during training but often struggle with novel vulnerabilities, unfamiliar attack techniques, or long-horizon reasoning. Second, outputs are inherently *non-deterministic*; small changes in prompts or code structure can produce substantially different responses. Consequently, single-run accuracy provides only a partial view of model performance, while benchmarks that ignore robustness to prompt or code variations are likely to overestimate capability [20].

Training-data contamination presents an additional challenge. Publicly released benchmark datasets may later appear in model training corpora, inflating benchmark scores without improving generalization. This problem is especially important in cybersecurity, where knowledge evolves rapidly. Distinguishing genuine reasoning from memorized knowledge has therefore become an important consideration in benchmark design.

Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) extends an LLM's parametric knowledge by incorporating evidence retrieved from external knowledge sources during inference [23]. Because cybersecurity knowledge changes continuously, retrieval enables systems to incorporate newly disclosed vulnerabilities, threat intelligence, and defensive guidance without retraining the underlying model.

RAG fundamentally changes evaluation because system performance depends on both the retriever and the language model. Retrieval failures occur when relevant evidence is not retrieved, whereas generation failures arise when retrieved evidence is ignored or interpreted incorrectly. Distinguishing these failure modes is essential for diagnosing system behaviour. Frameworks such as RAGAS [24] address this problem through reference-free metrics including faithfulness, answer relevance, and context relevance.

Retrieval also introduces an additional attack surface. By poisoning the external knowledge source, an adversary can manipulate generated responses without modifying model parameters. This threat has important implications for the design of trustworthy cybersecurity benchmarks and motivates the threat model presented in the following subsection.

Agentic Systems

Modern cybersecurity increasingly relies on LLM agents that combine language models with planning, memory, tool use, and iterative execution to perform complex tasks. Unlike single-turn interactions, these systems maintain state across multiple reasoning steps and adapt their behaviour based on intermediate observations, enabling more sophisticated offensive and defensive workflows.

Agentic architectures are now used for both automated penetration testing and defensive security operations. Systems such as PentestGPT maintain persistent execution state throughout penetration-testing sessions, while multi-agent frameworks have demonstrated the ability to discover and exploit both one-day and previously unknown vulnerabilities through planner–executor coordination [15,16]. Defensive agents similarly support threat intelligence,

Table 1: Threat model for LLM-in-the-loop security systems and the corresponding RIRAG safeguards.

Attack surface	Attacker capability	Effect on the evaluation or defense pipeline	RIRAG safeguard
Prompt manipulation	Controls part of the model input through direct jailbreaking or indirect prompt injection [6–8].	Hijacks the Evaluator Agent or another framework agent, bypasses safety controls, or corrupts evaluation judgments.	Continuous adversarial testing through the red-team loop (Section 6.1) and injection-aware prompt construction (Section 5.6).
Retrieval poisoning	Injects malicious or misleading documents into the indexed knowledge corpus [9,27].	Steers retrieval and generation without modifying model parameters and may corrupt the evidence treated as trusted context.	Validator consistency and confidence scoring, source-provenance hashing, quarantine support, and HITL review as a security boundary (Section 5.5).
Training poisoning	Introduces poisoned examples, labels, or corrections into the fine-tuning dataset [28].	Degrades model alignment, introduces persistent errors, or implants hidden behavior in retrained framework agents.	Human-validated labels, active-learning audit sampling (Section 6.4), training-data provenance, held-out gold evaluation, and rollback of regressive updates.

evidence validation, incident response, and benchmark evaluation.

Evaluating agentic systems is more challenging than evaluating conventional LLMs because performance depends on planning, memory, tool use, and recovery from intermediate failures in addition to reasoning quality. As errors may arise at any stage of a multi-step workflow, end-to-end success rates alone provide limited diagnostic value, making trajectory-level evaluation essential.

Threat Model

Evaluation frameworks built on LLMs, retrieval systems, and continuously updated external knowledge inherit the attack surfaces of each component. We consider an adversary $\mathcal{A}$ whose objective is to manipulate an LLM-assisted cybersecurity system—whether deployed operationally or used for benchmarking to produce incorrect, unsafe, or attacker-controlled outputs. Table **1** summarizes three principal attack surfaces.

(1) Inference-time prompt manipulation. The adversary controls part of the model input through *direct jailbreaking*, which attempts to bypass safety mechanisms [7,25], or *indirect prompt injection*, where malicious instructions are embedded within external content such as web pages, log files, or CVE descriptions [6,8]. Recent studies show that adaptive attackers routinely bypass static defenses, making continuous robustness assessment essential [26].

(2) Retrieval-corpus poisoning. Retrieval-Augmented Generation (RAG) introduces a second attack surface because manipulation of the knowledge corpus can directly influence retrieved evidence and model outputs. PoisonedRAG demonstrates that injecting only a small fraction of malicious documents can substantially increase attack success rates [9,27]. Since cybersecurity systems increasingly rely on continuously updated threat intelligence, retrieval validation becomes both a quality-assurance process and a security control.

(3) Training- and fine-tuning-time poisoning. Systems that periodically retrain models or agents remain vulnerable to poisoned labels and malicious fine-tuning data, which can weaken safety alignment or implant hidden backdoors [28]. Common mitigations include provenance tracking, careful dataset curation, human validation of high-impact samples, and controlled retraining procedures.

These observations motivate a guiding principle for cybersecurity benchmarking: an evaluation framework is only as trustworthy as its most vulnerable input channel. Consequently, trustworthy evaluation requires assessing not only the behaviour of the model under test but also the resilience of the surrounding retrieval, data-ingestion, and assessment infrastructure against adversarial manipulation.

Survey Method

The taxonomy presented in Section 3 was developed through a structured review of the cybersecurity and machine-learning literature published between 2023 and mid-2026. Sources included peer-reviewed publications from leading cybersecurity and machine-learning venues, influential preprints, industrial benchmark suites, and empirical studies evaluating LLMs in cybersecurity.

For each benchmark, we recorded its evaluation objective, task formulation, dataset provenance, methodology, metrics, and reported strengths and

limitations. Rather than comparing absolute leaderboard scores, which are sensitive to rapidly evolving model versions, we focused on recurring performance trends, evaluation methodologies, and documented failure modes.

The selected benchmarks provide representative coverage of the principal evaluation paradigms in LLM cybersecurity benchmarking. The resulting taxonomy emphasizes conceptual diversity over exhaustive cataloguing, enabling the identification of common design patterns, methodological trends, and the structural gaps examined throughout this survey.

Benchmark Taxonomy

We group LLM cybersecurity benchmarks into three categories: knowledge-oriented benchmarks that test foundational understanding, task-oriented benchmarks that measure applied security capabilities, and holistic frameworks that examine usability, operational risk, and deployment consequences. Table **2** summarizes representative work. Before presenting the categories, we make the construction of the taxonomy itself explicit: which benchmarks were included, along which dimension they are classified, why that dimension was chosen over alternatives, and how boundary cases are resolved.

Table 2: Comparison of Representative LLM Cybersecurity Benchmarks.

Benchmark	Focus	Tasks	Dataset / Method	Metrics	Main Strength / Limitation
CSEBenchmark	Expert knowledge	MCQ	Expert roadmaps and cognitive taxonomy [4]	Accuracy	Detailed taxonomy; limited specialist coverage.
SecEval	Foundational knowledge	MCQ	OWASP, MITRE, and GPT-4 generation [29]	Accuracy	Industry aligned; limited human validation.
CyberMetric	Broad knowledge	MCQ	RAG over NIST and RFCs; partial expert review [30]	Accuracy	Large scale with human baseline; weaker on emerging topics.
SecBench	Broad knowledge	MCQ, SAQ	Certification material; GPT-4 and expert review [31]	Accuracy	Large and multilingual; language bias and annotation errors.
CS-Eval	Comprehensive knowledge	MCQ, judgment, generation	42 categories; hybrid construction [32]	Accuracy, pass rate	Broad coverage; static content.
VulDetectBench	Vulnerability detection	Code analysis	Real CVEs and NIST SARD [33]	Accuracy, F1	Progressive task difficulty; limited to C/C++.
SecLLMHolmes	Robust vulnerability reasoning	Code analysis	Hand-crafted CVEs [20]	Accuracy	Tests reasoning stability; limited scale.
Purple Llama CyberSecEval	Secure coding	Code generation	MITRE ATT&CK and insecure-code patterns [34]	Compliance	Covers dual-use behavior; mainly code focused.
NYU CTF Bench	Offensive security	Interactive agent	Dockerized CSAW CTF challenges [35]	Pass@k	Supports live interaction; limited operational realism.
PentestGPT	Penetration testing	Interactive agent	Hack The Box and VulnHub [16]	Task completion	Realistic workflow; weak long-horizon context retention.
One-/Zero-day Studies	Autonomous exploitation	Interactive agent	Curated CVEs and planner–executor agents [14,15]	Success rate	Shows end-to-end exploitation; small curated datasets.
CTIBench	Threat intelligence	Multi-task	CTI reports and CVEs [5]	Accuracy	Early CTI-specific benchmark; uncertain ground truth.
CTIArena	CTI reasoning	Multi-task	Heterogeneous CTI sources [36]	Accuracy	Tests cross-source reasoning; complex construction.
CyberSOCEval	SOC operations	Malware, CTI	Operational SOC tasks [1]	Accuracy	High operational relevance; partly vendor specific.
SECURE	ICS security	MCQ, generation	Industrial-control-system advisories [37]	Accuracy	Covers critical infrastructure; narrow domain.
NetSPI Benchmark	Security vs. usability	Jailbreak	Adversarial prompts [38]	Jailbreak rate	Measures security–usability trade-off; limited methodological detail.
Cybench	Capability and risk	Interactive agent	Professional CTF tasks [39]	Success rate	Risk-oriented evaluation; constrained by CTF realism.
Risk Assessment Framework	Operational risk	Conceptual	Capability–adoption–impact model [21]	Capability, adoption, impact	Broad risk perspective; not executable.

Taxonomy Construction: Criteria and Boundary Cases

Inclusion and exclusion criteria: A benchmark or evaluation framework was included if it satisfies four conditions: (i) it was published or substantially updated between 2023 and mid-2026, the period in which LLM-specific cybersecurity evaluation emerged as a distinct practice; (ii) it evaluates LLMs or LLM-based agents on security-relevant inputs or tasks, rather than on general language ability; (iii) it defines a reproducible task formulation with explicit scoring criteria; and (iv) its public documentation is sufficient to determine dataset provenance, methodology, and metrics the attributes recorded in Section 2.5. We excluded general-purpose NLP benchmarks without security content (discussed only as background), red-teaming tools that lack a defined evaluation protocol or metric, and vendor capability claims published without reproducible methodology. Applying these criteria to the corpus described in Section 2.5 yielded the benchmarks analyzed in this section; Table **2** lists the representative set.

Classification dimension: Benchmarks are classified by the *primary evaluative claim* their headline metric validates: a claim about what a model *knows* (declarative security knowledge), about what it can *do* (procedural execution of a security task in a defined environment), or about how the *deployed system behaves* (usability, safety–utility balance, or operational risk of the model in context). This dimension follows the long-standing distinction between declarative and procedural knowledge in assessment theory and, more importantly, tracks the construct that a benchmark score is used as evidence for. It is analytically appropriate for three reasons. First, it is predictive: item format, dataset construction, metric family, and contamination exposure co-vary with claim type—knowledge claims are typically operationalized as static question banks scored by accuracy, task claims as interactive environments scored by success rate, and system-level claims as composite risk or usability measures. Second, it aligns with the failure modes this survey documents: the knowledge–reasoning gap of Section 4 is visible precisely at the boundary between the first two categories, and the operational gaps arise from the near absence of the third. Third, it is stable: a benchmark's primary claim does not change as model rankings shift, whereas orderings based on performance or popularity do.

Alternatives considered: Three alternative organizing dimensions were considered and rejected. Classification by security domain (vulnerability analysis, threat intelligence, SOC operations, and so on) cuts across evaluation design: a multiple-choice CTI quiz and an autonomous CTI extraction agent share a domain but almost nothing methodologically, so a domain taxonomy cannot expose the evaluation-design gaps this survey targets; we instead retain domain as a secondary facet in Table **2**. Classification by attack lifecycle phase (for example, ATT&CK tactics) fails to cover defensive and analytical benchmarks that have no natural lifecycle position, and many agentic benchmarks span several phases. Classification by metric family (accuracy, pass@k, success rate) is a surface property: several benchmarks report multiple metric families, and metrics follow from the claim being validated rather than the reverse.

Boundary-case resolution: Benchmarks that span categories are classified by the primary claim their headline metric validates. Purple Llama CyberSecEval, for example, involves security knowledge but its pass criteria depend on properties of generated code, so it is task-oriented. Cybench measures task success, yet its stated purpose and reporting—capability elicitation with risk framing and partial credit—make claims about deployment consequences, so we place it with the holistic frameworks. NetSPI's jailbreak evaluation measures a behavioral property of the deployed system (the security–usability balance) rather than knowledge or task skill, and is likewise holistic. SecBench combines multiple-choice and short-answer items, but both validate declarative knowledge, so it remains knowledge-oriented. When a benchmark offers several tracks, we classify it once, by the claim of its principal published results, and note secondary tracks in the analysis.

Knowledge Benchmarks

Knowledge-oriented benchmarks test factual, conceptual, and procedural cybersecurity understanding, usually through multiple-choice or short-answer questions.

CSEBenchmark organizes cybersecurity knowledge into factual, conceptual, and procedural categories derived from expert roadmaps [4]. Its results show a recurring pattern: LLMs perform relatively well on facts and concepts but struggle with procedures, specialist tools, and practical workflows.

Later benchmarks expanded scale and coverage. **SecEval** uses sources such as OWASP and MITRE to support industry-aligned evaluation [29]. **CyberMetric** applies retrieval-augmented question generation over

NIST publications and RFCs and includes a human baseline [30]. **SecBench** adds multilingual and short-answer evaluation [31], while **CS-Eval** combines multiple-choice, judgment, and generation tasks across 42 security categories [32].

These benchmarks provide broad coverage, but most remain static and place greater weight on recall than on applied reasoning. Frequent expert review could improve validity, although it also raises the cost of maintaining benchmarks in a rapidly changing field.

Task Benchmarks

Task-oriented benchmarks ask whether models can apply security knowledge to code, tools, reports, and interactive environments.

Vulnerability Analysis

VulDetectBench evaluates C/C++ vulnerability analysis using code from open-source projects and NIST SARD [33]. Its five tasks progress from binary vulnerability detection to identifying the root cause and trigger location. Models often detect that a flaw exists but perform substantially worse when precise semantic reasoning is required.

SecLLMHolmes examines whether vulnerability judgments remain stable under meaning-preserving changes, including variable renaming and irrelevant code insertion [20]. Small modifications can reverse model predictions, suggesting that some correct answers depend on superficial patterns rather than robust program understanding.

Purple Llama CyberSecEval evaluates both insecure-code generation and compliance with malicious requests grounded in MITRE ATT&CK [34]. It therefore connects defensive coding quality with the risk of offensive misuse, although its scope remains primarily code based.

Offensive Security

Interactive benchmarks require models to inspect environments, invoke tools, interpret results, and preserve context across several steps.

NYU CTF Bench provides 200 Dockerized CSAW Capture-the-Flag challenges in which agents can interact with live environments [35]. This moves evaluation beyond static question answering, but CTF settings still provide only a partial representation of real operations.

PentestGPT evaluates LLM support for penetration-testing workflows [16]. Models can often propose commands and interpret individual outputs, yet their performance declines when investigations require long-term context and consistent tracking of earlier evidence.

Cybench contains 40 professional CTF tasks divided into subtasks, allowing difficulty calibration and partial-credit measurement [39]. It also frames performance in terms of both capability and risk.

One-day and zero-day studies extend this evaluation to autonomous exploitation. Individual agents have exploited selected one-day vulnerabilities using public descriptions, while planner–executor teams have been tested on curated zero-day scenarios [14,15]. These studies are small and controlled. Even so, they shift the evaluation target from explaining an exploit to executing one through a complete sequence of actions.

Threat Intelligence and Security Operations

CTIBench evaluates tasks such as mapping CVE descriptions to CWE categories and extracting MITRE ATT&CK techniques from reports [5]. Its main limitations concern possible label inaccuracies and the restricted context provided by short CVE descriptions.

CTIArena draws on heterogeneous CTI sources and separates knowledge retrieval from reasoning [36]. This distinction matters because knowing an ATT&CK definition does not ensure that a model can infer the corresponding technique from a complex report.

CyberSOCEval targets malware analysis and higher-level threat-intelligence tasks intended to resemble SOC work [1]. Its operational focus improves realism, although some data remain vendor specific. **SECURE** concentrates on Industrial Control Systems using security advisories and related questions [37]. Its narrow scope addresses risks that general enterprise benchmarks may overlook.

Holistic Benchmarks

Recent frameworks examine whether benchmark success corresponds to safe and useful deployment.

The **NetSPI Open LLM Security Benchmark** measures jailbreak susceptibility alongside the effect of safeguards on legitimate use [38]. This captures an important trade-off: a model that refuses every request may appear secure but has little practical value.

Risk-based approaches argue that technical capability alone does not determine real-world harm [21]. Two additional factors matter. The first is adoption: whether an adversary would use the capability given its cost, reliability, and efficiency relative to existing tools. The second is impact: the damage that successful use could produce. Section 6.2 formalizes this relationship.

The benchmark landscape therefore reflects three successive questions. Does the model possess cybersecurity knowledge? Can it apply that knowledge to practical tasks? What risk follows when the capability is deployed? Knowledge benchmarks address the first question, interactive and task-specific benchmarks address the second, and emerging holistic frameworks address the third.

Evaluation Gaps and Design Requirements

The preceding taxonomy reveals a recurring performance limitation and several weaknesses in how LLM cybersecurity capabilities are evaluated. We summarize these as one central finding, three benchmark-construction flaws, and five gaps between benchmark results and operational risk. Table **3** links each gap to the corresponding RIRAG mechanism.

The Knowledge Reasoning Gap

Large proprietary models generally outperform open-source alternatives, although the gap is narrowing [2,30,35,37]. Rankings, however, reveal less than the shared failure pattern: models perform better on factual recall than on procedural and causal reasoning.

CSEBenchmark found strong performance on factual and conceptual questions but substantially weaker results on procedures involving specialist tools [4]. VulDetectBench reported a similar pattern in source-code analysis. Models often recognized that a vulnerability existed but struggled to identify its root cause [33]. SecLLMHolmes further showed that these judgments are brittle; meaning-preserving changes, including variable renaming, can reverse model predictions [20].

Current evidence therefore does not support replacing human experts with autonomous LLM agents in high-stakes security decisions [2,4,20]. A more defensible near-term role is that of a co-pilot: summarizing evidence, automating routine work, and drafting analyses for human review.

This knowledge–reasoning gap may partly reflect the distribution of training data. Declarative information what a concept is or why it matters is widely represented in text. Procedural knowledge is different. It depends on sequential action, tool feedback, changing context, and recovery from mistakes. These findings suggest that additional text alone may be insufficient; training and evaluation in interactive security environments may also be required.

Benchmark Design Flaws

Benchmark validity depends on how evaluation data are produced and maintained. Three construction problems recur.

Dataset provenance. Methods range from expert-designed roadmaps, as used by CSEBenchmark [4], to scraped vulnerability records and LLM-generated questions [30]. Expert construction improves relevance but scales poorly. Automated collection increases volume but can preserve source errors, noise, and bias.

Insufficient human validation. Reliable benchmark construction remains labor intensive. CSEBenchmark required 772 expert hours to validate

Table 3: Conceptual gaps between current benchmarks and operational risk, with the corresponding RIRAG mechanisms.

Gap	Benchmark Symptom	Consequence	RIRAG Mechanism
Capability vs. risk	Reports whether a model can perform a task but omits adoption and economic factors	Cannot estimate practical misuse risk	Operational risk score (§6.2)
Failure granularity	Reports one aggregate score for a multi-stage task	Provides little diagnostic value	Execution-stage error taxonomy (§5.6)
Human–machine teaming	Evaluates the model without an analyst	Does not reflect co-pilot deployment	Uplift and cognitive-load track (§6.3)
Staticity and contamination	Uses fixed, public evaluation items	Causes obsolescence and memorized scores	Living Knowledge Circuit and freshness windows (§5.5)
Adversarial robustness	Leaves retrieval and judging paths untested	Allows poisoned data or injected answers to corrupt results	Validation boundary and self-red-teaming (§6.1)

11,050 questions [4]. This effort illustrates the difficulty of detecting subtle factual, contextual, and logical errors. Reduced review lowers cost, but it also weakens confidence in the resulting scores.

Staticity. Most benchmarks are fixed snapshots. Their relevance declines as new vulnerabilities, tools, and attack methods appear. Public release also creates contamination risk: evaluation items may enter later training corpora, causing memorization to appear as generalization.

OPERATIONAL GAPS

Capability and Risk

Most benchmarks ask whether a model can perform a task. For example, NYU CTF Bench measures whether an agent can solve selected CTF challenges [35]. Such results establish capability, not operational risk.

Risk assessment must also consider whether an attacker would adopt the system. An LLM may offer little practical advantage if existing tools are faster, cheaper, or more reliable. Relevant measures include time-to-exploit, cost per operation, required expertise, and improvement over established methods. These economic and adoption factors are largely absent from current evaluations. Risk-oriented analysis [21] and NetSPI's security–usability evaluation [38] begin to address this problem. Section 6.2 makes these factors measurable.

Failure Diagnostics

A single score indicates whether a model failed but rarely explains the cause. In a multi-stage task, failure may result from missing knowledge, incorrect reasoning, prompt misinterpretation, poor tool use, or loss of earlier context.

CSEBenchmark provides fine-grained knowledge categories [4], but comparable diagnostics remain uncommon in interactive benchmarks. Developers need to know where an execution failed, not only whether the final answer was incorrect. RIRAG therefore records errors by category and execution stage.

Human-AI Teaming

Near-term deployments are likely to use LLMs as assistants rather than independent decision makers [1,40]. Most benchmarks nevertheless evaluate models in isolation.

This creates a mismatch between testing and use. The relevant question is not simply whether the model is accurate, but whether it improves the performance of a human–machine team. Useful measures include analyst decision accuracy, time-to-resolution, cognitive load, and the frequency with which human reviewers detect or correct model errors. Field studies have begun to examine SOC collaboration [40], but controlled teaming benchmarks remain limited. Section 6.3 introduces such an evaluation track.

Staticity and Contamination

Static datasets create both temporal and statistical validity problems. First, cybersecurity knowledge changes quickly. Strong performance on older vulnerabilities may not predict behavior on recent or unseen threats. Second, publicly available test items can enter model training data, making it difficult to distinguish reasoning from memorization.

A meaningful evaluation must therefore control item age, exposure, and freshness. It should also introduce new tasks after model training whenever possible. The Living Knowledge Circuit in Section 5.5 addresses this requirement through freshness windows and continuously updated evaluation material.

Benchmark Robustness

Benchmarks are themselves attack surfaces. Two channels are especially important.

The first is corpus poisoning. Systems that use retrieval-augmented generation can be influenced by malicious documents placed in the retrieval collection. PoisonedRAG demonstrates that poisoning a small part of a corpus can disproportionately affect retrieval outcomes [9].

The second is judge manipulation. An LLM used to score candidate answers may follow instructions embedded within the answer being evaluated. Static safeguards are also vulnerable to adaptive attacks [26]. A compromised retrieval or judging pipeline can silently distort benchmark results.

Evaluation infrastructure must therefore treat data ingestion, retrieval, and scoring as security boundaries. RIRAG addresses these risks through validation controls and continuous adversarial testing in Sections 5.5 and 6.1.

Operational Implications

Commercial security products are adopting LLMs despite the limitations of current evaluation practices

[2]. A high knowledge score may conceal weak procedural reasoning, sensitivity to irrelevant input changes, or poor performance across long task sequences [4,20]. Such results can create unjustified confidence in systems used for consequential decisions.

The problem is operational, not merely methodological. Attackers may deliberately target the same weaknesses that benchmarks overlook. Closing these gaps is therefore necessary for credible risk assessment and safe deployment.

From Gaps to Design Requirements

Table **3** summarizes the correspondence between gaps and RIRAG mechanisms; this subsection makes the correspondence derivational rather than declarative. For each gap we state the requirement that any evaluation framework must satisfy to close it, show why the RIRAG mechanism follows from that requirement under the constraints of Section 2.4, and record the alternative designs that were considered and rejected. The requirements R1–R5 defined here are the premises from which the design principles of Section 5.2 are subsequently derived.

G1: capability versus risk ⇒ R1. If benchmark output is to inform deployment and misuse decisions, the reported quantity must be monotone in the factors that determine operational consequence—capability, adversary adoption, and impact—not in capability alone. Reporting capability alongside a qualitative risk narrative was rejected because narratives are not comparable across models and invite motivated interpretation. An additive combination of the three factors was rejected because it assigns non-zero risk to systems with zero capability or zero adoption; a multiplicative form (Eq. **16**) preserves the property that each factor is individually necessary. Full probabilistic attack simulation would be more expressive but was rejected as a core mechanism because its cost and sensitivity to threat-model assumptions make it unsuitable for routine benchmarking; it remains a calibration path for the risk model (Section 11).

G2: failure granularity ⇒ R2. Diagnosing multi-stage failures requires that scores be decomposable by execution stage, which is only possible if the pipeline is instrumented at stage boundaries. This entails structured execution logging and an error taxonomy (Section 7.6) rather than any post-hoc analysis. Asking an LLM judge to explain failures after the fact was rejected: the explanation inherits the judge's own reliability problems and violates the separation of judge and subject (P4). Purely manual error analysis was rejected as the primary mechanism because it does not scale with continuous evaluation, though it is retained inside the human-review band.

G3: human–AI teaming ⇒ R3. If deployed use is assistive, the unit of analysis must include the analyst; no property measured on the model alone can establish complementarity. This entails a controlled, within-subjects teaming track (Section 6.3). Proxy measures on the model alone, such as explanation quality, were rejected because the human-factors literature shows they do not predict team performance. Observational SOC telemetry was rejected as the primary instrument because assignment is not controlled and confounds cannot be excluded; it is retained as an ecological-validity complement.

G4: staticity and contamination ⇒ R4. If item exposure and item age affect scores, a valid benchmark must control them as variables, which requires that the item supply be continuous rather than episodic. This entails the living Knowledge Circuit with freshness windows and a frozen comparability partition (Section 5.5). Periodic manual refresh was rejected because each refresh interval is itself a contamination window and expert effort arrives in unsustainable bursts. Fully synthetic item generation was rejected as the sole supply because provenance and construct validity cannot be established for items with no operational grounding. A private static test set was rejected because privacy does not stop aging, and a single leak destroys the instrument.

G5: benchmark robustness ⇒ R5. Because the threat model of Section 2.4 places the evaluation pipeline itself inside the attack surface, evaluation infrastructure must be defended and tested like any other security system. This entails validation boundaries on ingestion, human review as a trust boundary, and continuous self-red-teaming (Section 6.1). Trust-on-ingest with source allowlists was rejected because allowlisted feeds can themselves be compromised, and corpus-poisoning attacks operate within trusted channels. A one-time security audit was rejected because adaptive attackers respond to fixed defenses; robustness must be measured continuously against an evolving attack archive.

Each requirement thus admits a small space of candidate designs, and in each case the selected mechanism is the one that satisfies the requirement without violating another. This is the sense in which the architecture of Section 5 is derived from, rather than merely associated with, the gaps documented above.

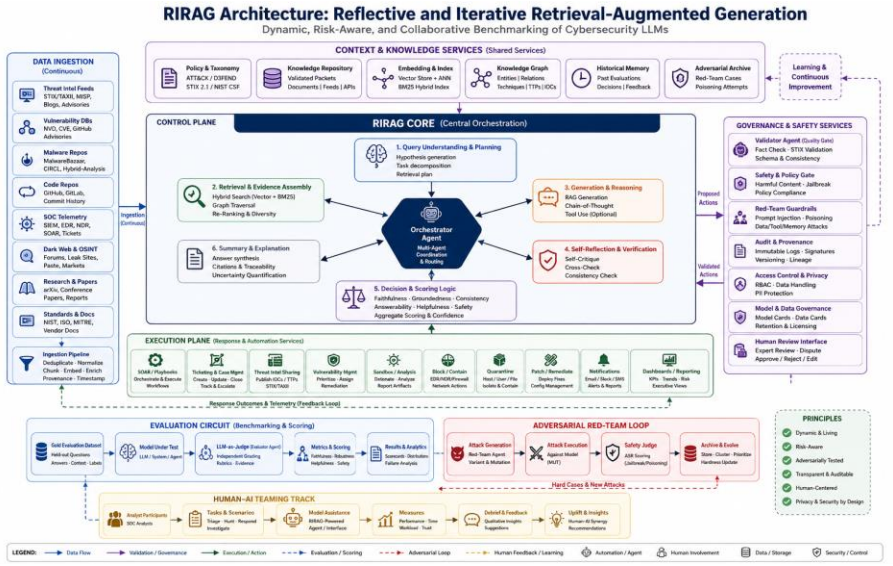

Figure 1: End-to-end architecture of the Reflective and Iterative Retrieval-Augmented Generation (RIRAG) framework. Continuous cybersecurity data ingestion feeds shared context and knowledge services used by the centrally orchestrated RIRAG core. Governance and safety services validate framework decisions, while the Evaluation Circuit, adversarial red-team loop, and human–AI teaming track provide complementary assessments of model capability, robustness, operational risk, and collaborative effectiveness. Evaluation outcomes and human feedback close the framework's continuous-improvement loops.

The RIRAG Framework

We propose Reflective and Iterative Retrieval-Augmented Generation (RIRAG) to address the limitations identified in Section 4. RIRAG replaces the conventional one-shot benchmark with a continuous, feedback-driven workflow. The framework evaluates newly ingested cybersecurity intelligence, records intermediate validation and evaluation decisions, and uses observed failures to improve both its knowledge resources and internal agents.

As illustrated in Fig. **1**, RIRAG integrates continuous intelligence ingestion, shared knowledge services, central agent orchestration, governance controls, model evaluation, adversarial red teaming, and human–AI teaming within a unified architecture. The Knowledge Circuit maintains the validated security knowledge used by the framework, while the Evaluation Circuit measures the retrieval, reasoning, generation, robustness, and operational characteristics of submitted models. Feedback from evaluation, expert review, and adversarial testing is returned to the knowledge and training processes, enabling the benchmark to evolve with the threat landscape.

Theoretical Foundations

The design principles presented in the next subsection are not free-standing engineering intuitions; they instantiate four established bodies of evaluation and security theory, applied to the requirements R1–R5 derived in Section 4.5.

The first is measurement theory and construct validity [41,42]. A benchmark score is not the quantity of interest; it is evidence for an inference from observed performance to an underlying construct, such as "vulnerability-analysis capability." The two canonical threats to that inference are construct under-representation and construct-irrelevant variance [42]. Both appear in the gaps of Section 4: static datasets under-represent a temporally evolving construct (G4), and contamination introduces construct-irrelevant variance by rewarding memorization over reasoning. Liveness (P1) and the freshness controls of the Knowledge Circuit are therefore validity requirements, not conveniences, and the diagnostic decomposition demanded by P5 follows from the measurement-theoretic observation that a single aggregate score cannot support decisions about *why* a system failed.

The second is the dynamics of *high-stakes quantitative indicators*, summarized in Campbell's law: the more a quantitative social indicator is used for decision-making, the more subject it becomes to corruption pressures that distort the process it is intended to monitor [43]. Benchmark leaderboards are precisely such indicators. Campbell's law predicts benchmark overfitting and contamination as systemic phenomena rather than accidents, and entails two design responses: continuously renewed, partially

held-out evaluation material (P1), and structural separation between the optimizing system and the measuring system (P4). The latter is the same independence argument that underlies financial auditing—the judge must not share optimization pressure with the subject it grades.

The third is *security engineering and risk theory*. Treating ingestion, retrieval, and judging as security boundaries (P3) applies the classical principles of complete mediation and defense in depth [44] to evaluation infrastructure, in direct response to R5. The operational risk model of Section 6.2 follows the standard decomposition of risk into threat likelihood and impact used in established risk-assessment methodology [45], with adversary adoption playing the role of threat likelihood; P6 is this decomposition restated as a design commitment.

The fourth is *signal quality and human-factors theory*. Reflection (P2) is uncertainty propagation: each stage of a measurement chain should emit calibrated confidence rather than a bare point decision, so that downstream consumers can weight evidence appropriately. Adversarial self-testing (P7) operationalizes the finding that defenses evaluated only against fixed attacks systematically overstate robustness against adaptive adversaries [26]. Finally, the teaming track of Section 6.3 rests on empirical human–AI interaction results showing that model accuracy does not by itself predict team performance, and that trust and reliance require deliberate measurement [46,47].

Together these foundations give each principle a theoretical warrant and a traceable origin: Gi ⇒ Ri (Section 4.5) ⇒ principle ⇒ mechanism. The architecture that follows should be read as the constructive proof that the requirements can be satisfied simultaneously.

Design Principles

Seven principles guide the framework. Each corresponds to one or more of the requirements derived in Section 4.5 and rests on the theoretical foundations above: P1 discharges R4, P2 and P5 discharge R2, P3, P4, and P7 discharge R5, P6 discharges R1, and the teaming track of Section 6.3 discharges R3.

P1 Liveness. The benchmark continuously incorporates recent cybersecurity intelligence so that its content remains aligned with current threats and is less dependent on fixed public datasets.

P2 Reflection. Each stage produces an explicit assessment, such as a confidence score, validation status, or diagnostic label. Later stages use this information rather than treating intermediate outputs as inherently reliable.

P3 Human-in-the-loop as a security boundary. Human experts validate benchmark content and review uncertain or suspicious inputs. They act as a trust boundary against inaccurate, manipulated, or poisoned data, not merely as a final quality check.

P4 Separation of judge and subject. The Evaluator Agent belongs to the framework and remains separate from the model under test. A submitted model therefore cannot directly grade its own output.

P5 Diagnostic granularity. The framework records results for individual tasks, execution stages, and error categories. Failures can therefore be traced to specific causes rather than represented only by an aggregate score.

P6 Risk awareness. Capability measurements can be combined with adversary adoption and impact estimates to produce the operational risk score defined in Section 6.2.

P7 Adversarial self-testing. RIRAG repeatedly tests its own ingestion, retrieval, and evaluation paths, as well as the models being assessed. Robustness is measured over time rather than assumed from a single evaluation; Section 6.1 develops this process.

Coupled Circuits

RIRAG consists of two connected feedback systems. The **Knowledge Circuit** described in Section 5.5 ingests and validates cybersecurity intelligence, updates the indexed knowledge base, and produces training data for the framework's internal agents. The **Evaluation Circuit** in Section 5.6 tests externally submitted models on cybersecurity tasks supported by that knowledge.

Both circuits use the same human-validated, STIX-structured source of truth but serve different purposes. The Knowledge Circuit updates what the framework can learn from; the Evaluation Circuit determines what a submitted model can retrieve, reason about, and generate.

Notation

We define the notation used in Sections 5.5–6.4. Let $\mathcal{K} = c_1, \ldots, c_N$ denote an indexed knowledge base containing N text chunks. Each chunk c_i is

represented by an embedding $\mathbf{e}_i = \phi(c_i) \in \mathbb{R}^d$, where ϕ is a fixed embedding model.

A *Knowledge Packet* is defined as

$$p = \langle \mathrm{id}, m, r, T, S, v \rangle. \quad (1)$$

Here, id is a unique identifier, m contains the source, URI, timestamp, and content hash, and r is the raw content. The set $T = t_1, \ldots, t_{|T|}$ contains training representations produced by the Translator Agent, including question–answer pairs, true/false statements, and summaries. The term S denotes the validated STIX 2.1 bundle. The record v contains the confidence score $\gamma \in [0,1]$, validation status, and human feedback.

The Translator, Validator, and Evaluator agents are denoted by Φ_{tr}, Φ_{val}, and Φ_{ev}, respectively. Each agent uses a task-specific prompt and output schema and may optionally be fine-tuned.

For a query q, the similarity between the query and chunk c_i is

$$s_i(q) = \frac{\phi(q) \cdot \mathbf{e}_i}{|\phi(q)|_2, |\mathbf{e}_i|_2}. \quad (2)$$

The retrieval operation is written as

$$\mathcal{R} * K(q) = \mathrm{TopK} *, i = 1, \ldots, N; s_i(q). \quad (3)$$

The notation $\mathcal{R}_K(q)$ refers to the K chunks associated with the highest similarity scores.

The Golden Evaluation Dataset is

$$\mathcal{D} * g = \left(q_j, a_j^\star, C_j^\star\right) * j = 1^M. \quad (4)$$

Here, q_j is an evaluation query, $a_j^\star$ is an expert-verified reference answer, and $C_j^\star \subseteq \mathcal{K}$ is the optional set of knowledge chunks needed to derive the answer.

A model under test is represented by g_θ. Given query q_j and retrieved context C_j, it produces

$$\hat{a} * j = g * \theta\left(q_j, C_j\right). \quad (5)$$

The generated answer $\hat{a}_j$ may also include citations to retrieved sources. Appendix A summarizes the notation used throughout the framework.

The Knowledge Circuit

The Knowledge Circuit continuously ingests, transforms, validates, and indexes cybersecurity intelligence. It also converts reviewed packets into training examples for the framework's internal agents. Two feedback loops support this process. The *knowledge loop* updates the indexed knowledge base, while the *training loop* uses validated corrections to refine the Translator and Validator agents. Algorithm **1** summarizes the workflow.

Algorithm 1 Knowledge Circuit Master Loop

Require: sources Σ, knowledge base $\mathcal{K}$
Require: agents Φ_{tr} and Φ_{val}
Require: thresholds τ_{pass}, τ_{fail}, and β
while true **do**
  $Q \leftarrow$ Ingest(Σ)
  $A \leftarrow$ AuditSample(Q)
  for all $p \in Q$ **do**
    $p.T \leftarrow \Phi_{\mathrm{tr}}(p.r)$
    $(p.S, \gamma) \leftarrow \Phi_{\mathrm{val}}(p.r, \mathcal{K})$
    $p.v.\gamma \leftarrow \gamma$
    $p.v.status \leftarrow \mathcal{T}(\gamma)$
    if $p.v.status \neq$ pass or $p \in A$ **then**
      $p \leftarrow$ HumanReview(p)
    else
      $p.v.status \leftarrow$ approved
    end if
    if $p.v.status =$ approved **then**
      Index($\mathcal{K}$, $p.S$)
      AppendTrainingSet(p)
    end if
  end for
  if $|\mathcal{D}_{\mathrm{tr}}| \geq \beta$ or an update is scheduled **then**
    $(\Phi_{\mathrm{tr}}, \Phi_{\mathrm{val}}) \leftarrow$ Retrain($\mathcal{D}_{\mathrm{tr}}$)
  end if
end while

Ingestion

The first phase acquires intelligence from heterogeneous sources, including National Vulnerability Database feeds, internal SOC playbooks, incident reports, and public threat-intelligence publications. Inputs may be structured, semi-structured, or unstructured, such as CVE records in JSON, threat reports in HTML, and playbooks in PDF form.

Incoming data are placed in a raw-data store or message queue. This buffer separates data collection from later processing and allows packets to be handled independently of their source format or arrival rate.

The ingestion path is treated as untrusted under the threat model in Section 2.4. Each packet records its source, URI, ingestion time, and a SHA-256 hash of the original content. These records support provenance tracing, duplicate detection, source quarantine, and revocation of knowledge derived from a compromised source.

Transformation

The Translator Agent Φ_{tr} converts raw content into standardized training representations. Its outputs may include question–answer pairs, true/false statements,

summaries, Indicators of Compromise, and Tactics, Techniques, and Procedures mapped to MITRE ATT&CK.

Algorithm **2** applies a self-consistency check. The agent generates several candidates for each output format and retains an item only when the samples agree sufficiently. This reduces the chance that unstable or hallucinated content enters the training and evaluation pipelines.

Algorithm 2 Translator Agent

Require: raw content r, prompt template π_{tr}
Require: sample count n_s
Ensure: training representations T
$T \leftarrow \emptyset$
for all $f \in \{\text{QA}, \text{TF}, \text{SUM}, \text{IOC}\}$ **do**
  $O \leftarrow \emptyset$
  for $k \leftarrow 1$ **to** n_s **do**
    $o_k \leftarrow \Phi_{\text{tr}}(\pi_{\text{tr}}(r, f))$
    $O \leftarrow O \cup \{o_k\}$
  end for
  $o^* \leftarrow \textsc{ConsistentOutput}(O)$
  if $o^* \neq \perp$ **then**
    $T \leftarrow T \cup \{o^*\}$
  end if
end for
return T

Validation

The Validator Agent Φ_{val} checks incoming content against the indexed knowledge base and produces a STIX 2.1 representation. It retrieves related knowledge, identifies conflicts, estimates confidence, and maps security entities and relationships into a machine-readable format.

For raw content r, let $C = \mathcal{R}_K(r)$ denote the K chunks returned by the retrieval operator in Section 5.4. The Validator computes three scores in $[0,1]$: groundedness γ_g, consistency γ_c, and novelty/coherence γ_n. Their weighted geometric mean is

$$\gamma = \gamma_g^{w_g} \gamma_c^{w_c} \gamma_n^{w_n}, \qquad w_g + w_c + w_n = 1. \tag{6}$$

The geometric mean is conservative: a near-zero score on any dimension sharply reduces the combined confidence. Triage is defined as

$$\mathcal{T}(\gamma) = \begin{cases} \text{pass}, & \gamma \geq \tau_{\text{pass}}, \\ \text{review}, & \tau_{\text{fail}} \leq \gamma < \tau_{\text{pass}}, \\ \text{fail}, & \gamma < \tau_{\text{fail}}. \end{cases} \tag{7}$$

The thresholds control the trade-off between review cost and the risk of accepting inaccurate knowledge. Section 6.4 describes how they may be adjusted using observed review outcomes.

Algorithm 3 Validator Agent

Require: raw content r, knowledge base $\mathcal{K}$
Require: weights (w_g, w_c, w_n)
Ensure: STIX bundle S, confidence γ, and status z
$C \leftarrow \mathcal{R}_K(r)$
$S \leftarrow \Phi_{\text{val}}(r, C)$
$\gamma_g \leftarrow \textsc{Groundedness}(S, C)$
$\gamma_c \leftarrow 1 - \max_{c \in C} \textsc{Conflict}(S, c)$
$\gamma_n \leftarrow \textsc{NoveltyCoherence}(S, C)$
$\gamma \leftarrow \gamma_g^{w_g} \cdot \gamma_c^{w_c} \cdot \gamma_n^{w_n}$
$z \leftarrow \mathcal{T}(\gamma)$
return (S, γ, z)

Human Review

Packets assigned review or fail are sent to cybersecurity experts. A sample of high-confidence pass packets is also reviewed to estimate false-acceptance rates and identify systematic Validator errors.

Reviewers may correct the Translator output, revise the STIX mapping, and approve or reject each packet. Their corrections and decisions are stored in the validation record and later used as supervised training examples. Audit sampling is necessary because a Validator that repeatedly accepts material from a poisoned source may otherwise appear reliable when its decisions never enter the review queue.

Feedback Loops

Approved packets close two related feedback loops.

Knowledge loop. The approved STIX bundle is chunked, embedded using ϕ, and inserted into the knowledge base. The new content then becomes available to later Validator queries and external retrieval-augmented systems.

Training loop. The original content, corrected training representations, and approved STIX bundle are added to a supervised dataset $\mathcal{D}_{\text{tr}}$. Once the dataset reaches threshold β, or when an update is scheduled, candidate versions of the Translator and Validator are fine-tuned and evaluated before deployment.

Algorithm 4 Dual-Closing Update

Require: approved packet p, knowledge base $\mathcal{K}$
Require: training set $\mathcal{D}_{\text{tr}}$ and threshold β
$S^{\dagger} \leftarrow \textsc{SelectSTIX}(p)$
$\mathcal{T}^{\dagger} \leftarrow \textsc{SelectFormats}(p)$
for all $c \in \textsc{Chunk}(S^{\dagger})$ **do**
  $\mathcal{K} \leftarrow \mathcal{K} \cup \{(c, \phi(c))\}$
end for
$\mathcal{D}_{\text{tr}} \leftarrow \mathcal{D}_{\text{tr}} \cup \{(p, r, \mathcal{T}^{\dagger}, S^{\dagger})\}$
if $|\mathcal{D}_{\text{tr}}| \geq \beta$ **then**
  $\Phi'_{\text{tr}} \leftarrow \textsc{FineTuneTranslator}(\Phi_{\text{tr}}, \mathcal{D}_{\text{tr}})$
  $\Phi'_{\text{val}} \leftarrow \textsc{FineTuneValidator}(\Phi_{\text{val}}, \mathcal{D}_{\text{tr}})$
  $u \leftarrow \textsc{GoldEval}(\Phi'_{\text{tr}}, \Phi'_{\text{val}})$
  if $u =$ pass **then**
    $\Phi_{\text{tr}} \leftarrow \Phi'_{\text{tr}}$
    $\Phi_{\text{val}} \leftarrow \Phi'_{\text{val}}$
  else
    discard candidate updates
  end if
end if

Each candidate update is tested on a held-out, human-curated dataset before deployment. An update that reduces performance or violates a safety requirement is rejected. This gate limits the damage that could result from erroneous or poisoned labels.

Let h_t denote the fraction of packets routed to human reviewers during epoch t. Under a stable source distribution, retraining may reduce h_t as the agents learn from previous corrections. A sudden increase may instead indicate distribution shift, a new threat class, model degradation, or source manipulation. The review rate is therefore both a cost measure and a monitoring signal.

Packet Schema

A Knowledge Packet provides the common representation used across all phases. Its main fields are packet_id, ingestion metadata, raw content, ai_training_formats, ai_structured_analysis, and validation data. Together, these fields support provenance tracking, debugging, revocation, and communication between independently implemented components. The field names follow the STIX 2.1 content model introduced in Section 5.4.

Complexity Analysis

Consider an epoch containing B packets and a knowledge base with N indexed chunks. Ingestion requires $O(B)$ packet operations in addition to reading and hashing raw bytes. Translation requires $O(Bn_s)$ model generations when the number of output formats is treated as constant.

Let $T_{\text{ANN}}(N)$ denote the cost of one approximate-nearest-neighbor lookup. Validation requires $O\big(BT_{\text{ANN}}(N)\big)$ retrieval work and $O(B)$ Validator calls. If an approved packet produces κ chunks, indexing requires $O\big(\kappa T_{\text{ins}}(N)\big)$, where $T_{\text{ins}}(N)$ is the vector-index insertion cost.

If a fraction h_t of the batch is reviewed and the mean review time is $\bar{\rho}$, the expected expert effort is

$$H_t = h_t B \bar{\rho}. \tag{8}$$

Retraining occurs only when enough new examples accumulate or when a scheduled update is triggered. Its cost depends on model size, tuning method, and the number of examples in $\mathcal{D}_{\text{tr}}$.

Table 4: Per-epoch cost of the Knowledge Circuit.

Phase	Cost	Resource
Ingestion	$O(B)$ packet operations	I/O
Translation	$O(Bn_s)$ generations	GPU
Validation	$O\big(BT_{\text{ANN}}(N)\big)$ plus $O(B)$ calls	Index/GPU
HITL	$h_t B \bar{\rho}$	Experts
Indexing	$O\big(\kappa T_{\text{ins}}(N)\big)$ per packet	Vector DB
Retraining	Periodic; model dependent	GPU

Human review is likely to remain the main operational bottleneck. The training loop is designed to reduce avoidable review over time, while audit sampling preserves independent oversight of high-confidence decisions.

Evaluation Circuit

The Evaluation Circuit benchmarks submitted retrieval-augmented generation systems against the validated knowledge produced by the Knowledge Circuit. It measures answer quality, source use, retrieval effectiveness, efficiency, and statistical reliability under a common execution protocol. The

framework supports two tracks. A *controlled-retrieval track* supplies every model with the same retrieved context, isolating generation quality. An *end-to-end track* allows each submitted system to retrieve its own context, thereby evaluating the complete RAG pipeline. Algorithm **5** summarizes the evaluation loop.

Algorithm 5 Evaluation Circuit for submitted model g_θ

Require: model g_θ, gold set $\mathcal{D}_g$, knowledge base $\mathcal{K}$
Require: evaluator Φ_{ev}, retrieval depth K, runs R
HealthCheck(g_θ)
$\mathcal{E} \leftarrow \emptyset$
for all $(q_j, a_j^\star, C_j^\star) \in \mathcal{D}_g$ **do**
  for $r = 1$ **to** R **do**
    $C_{jr} \leftarrow \mathcal{R}_{\mathcal{K}}(q_j)$
    $u_{jr} \leftarrow$ BuildPrompt(q_j, C_{jr})
    $(\hat{a}_{jr}, S_{jr}) \leftarrow g_\theta(u_{jr})$
    $s_{jr} \leftarrow \Phi_{\text{ev}}(q_j, C_{jr}, \hat{a}_{jr}, a_j^\star)$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(j, r, s_{jr}, \ell_{jr})\}$
  end for
end for
return Aggregate$(\mathcal{E})$

Model Integration

Each submitted model is packaged as a Docker container that exposes a fixed REST endpoint, such as `POST /query`, and follows a common JSON contract. The request contains a query and, in the controlled-retrieval track, the context selected from $\mathcal{K}$. The response contains a generated answer and may include cited source identifiers. Before evaluation, the framework checks availability, response format, latency limits, and resource constraints.

Containers execute in a restricted environment with no access to gold answers, evaluator prompts, or internal services. This design improves reproducibility and limits the effect of a faulty or adversarial submission.

Golden Dataset

The Golden Evaluation Dataset is $\mathcal{D}_g = \{(q_j, a_j^\star, C_j^\star)\}_{j=1}^{M}$, as defined in Section 5.4. It contains expert-verified cybersecurity queries, reference answers, and optional sets of essential knowledge chunks. The queries cover areas such as vulnerability analysis, threat-actor behavior, and incident response.

The gold set is held out from the training data produced by the Knowledge Circuit. To reduce contamination, part of the set is private and periodically replaced with tasks derived from recently validated intelligence. Public and private subsets are reported separately.

RAG Execution

In the controlled-retrieval track, the framework embeds each query using the same model ϕ used to index $\mathcal{K}$ and retrieves the top-K chunks through $\mathcal{R}_K$. It then constructs a prompt containing the system instruction, delimited context, and query. The same context and prompt template are used for every submitted model.

In the end-to-end track, the container receives the query and performs its own retrieval. It must return the generated answer and the identifiers of the sources used. This track measures retrieval and generation jointly, whereas the controlled track isolates generation quality.

Retrieved text is treated as untrusted data. Context blocks are clearly delimited, and the system instruction prohibits following commands embedded in the retrieved content. The framework records the answer, cited sources, latency, token use, and execution status for every run.

Automated Scoring

The Evaluator Agent Φ_{ev} is part of the benchmark and remains separate from the submitted model. It receives the query q, retrieved context C, generated answer $\hat{a}$, and reference answer $a^\star$. It then computes the metrics in Table **5**. All normalized quality metrics lie in $[0,1]$, with larger values indicating better performance.

Let $\mathrm{Cl}(\hat{a}) = \{x_1, \ldots, x_n\}$ be the atomic claims extracted from an answer, following the claim-based evaluation used by RAGAS [24]. Faithfulness is

$$F(\hat{a}, C) = \frac{|\{x \in \mathrm{Cl}(\hat{a}) : C \vDash x\}|}{|\mathrm{Cl}(\hat{a})|}. \tag{9}$$

Here, $C \vDash x$ means that the retrieved context supports claim x. When an answer contains no extractable claims, the case is flagged for review rather than assigned an arbitrary score.

For answer relevance, the evaluator generates n_q questions $q'_1, \ldots, q'_{n_q}$ from $\hat{a}$. The score is

$$A(\hat{a}, q) = \frac{1}{n_q} \sum_{i=1}^{n_q} \mathrm{sim}\left(\phi(q), \phi(q'_i)\right). \tag{10}$$

When gold context $C^\star$ is available, context precision and recall are

$$P_C = \frac{|C \cap C^\star|}{|C|}, \qquad R_C = \frac{|C \cap C^\star|}{|C^\star|}. \tag{11}$$

These metrics apply directly to the end-to-end track. In the controlled-retrieval track, they characterize the shared framework retriever and are not used to distinguish submitted generators.

Factual correctness is measured by claim-level F1:

$$F_{\text{fact}} = \frac{2\,\text{TP}}{2\,\text{TP} + \text{FP} + \text{FN}}. \tag{12}$$

The benchmark may report a composite score

$$S = \sum_{k=1}^{L} \lambda_k\, m_k, \qquad \sum_{k=1}^{L} \lambda_k = 1, \quad \lambda_k \geq 0, \tag{13}$$

where m_k denotes a metric and λ_k its declared weight. Individual metrics remain visible so that the composite score does not hide important trade-offs.

Table 5: Evaluation metrics for RAG systems.

Type	Metric	Purpose
Retrieval	Context precision	Relevant retrieved content
Retrieval	Context recall	Required content retrieved
Generation	Faithfulness	Support from retrieved context
Generation	Answer relevance	Direct response to the query
Generation	Factual correctness	Agreement with reference claims
Efficiency	Latency	Response time
Efficiency	Token use	Input and output cost

Candidate answers are passed to Φ_{ev} as quoted data rather than executable instructions. Human reviewers audit a sample of judgments, with higher sampling rates for low-confidence or anomalous cases. This limits, but does not eliminate, prompt injection and judge-model bias.

Reporting and Statistics

The final report provides per-metric means, domain-level breakdowns, latency, token use, failure categories, and selected examples. Raw per-query scores are retained for debugging and paired comparisons. The leaderboard identifies the evaluation track, prompt version, knowledge-base snapshot, model configuration, and metric weights used for each result.

Because model outputs are stochastic, each query is evaluated over R independent runs. Aggregate results include bootstrap 95% confidence intervals over queries and runs. Pairwise claims of superiority are supported by paired bootstrap tests at $\alpha = 0.05$, with correction for multiple comparisons. This prevents small sampling fluctuations from being presented as meaningful differences between models.

FRAMEWORK EXTENSIONS

The core framework of Section 5 is extended along four axes: continuous adversarial red-teaming and an operational risk model, which discharge requirements R5 and R1; a human–AI teaming track, which discharges R3; and graph-enhanced retrieval with active-learning review, which scales the knowledge and review processes that requirement R4 makes continuous. Each extension is specified below.

Adversarial Red-Teaming

The base RIRAG framework evaluates whether a model answers cybersecurity queries accurately and faithfully. It does not directly measure behavior under adversarial pressure. To address this limitation, RIRAG adds a Red-Team Agent Φ_{rt} that continuously generates and evaluates adversarial cases. This extension implements principle P7: robustness is measured repeatedly because adaptive attacks can bypass fixed defenses [26].

Scope

The red-team loop targets three attack surfaces. First, the model under test may be vulnerable to jailbreaks, indirect prompt injection, or meaning-preserving perturbations [20,25,34]. Second, an attacker may poison the retrieval corpus and influence which evidence reaches the model [9]. Third, an LLM-based evaluator may be manipulated by instructions embedded in the candidate answer. The Red-Team Agent tests the submitted model, retrieval corpus, and evaluator separately so that failures can be attributed to the affected component.

Attack Archive

The agent maintains an attack archive $\mathcal{L}$ organized by risk category and attack style, following the quality–diversity principle used in Rainbow Teaming [48]. Risk categories may include malicious-code generation, data exfiltration, or evasion advice. Attack styles may include role-play, obfuscation, multilingual prompts, and indirect injection. Each archive cell stores the

highest-scoring attack found for one category–style pair. A generator mutates archived attacks, while a separate safety judge Φ_{safe} assigns an attack-success score in $[0,1]$.

The same procedure can test corpus poisoning or evaluator injection. Only the injection point and success criterion change. Corpus attacks are scored by retrieval influence, while evaluator attacks are scored by whether they change the assigned judgment.

Algorithm 6 Adversarial Red-Team Loop

Require: target g_θ, generator Φ_{rt}
Require: safety judge Φ_{safe}, archive $\mathcal{L}$
Require: attack threshold τ_{atk}, budget G
Ensure: updated archive $\mathcal{L}$, robustness $\hat{\rho}$
$n_{\text{succ}} \leftarrow 0$
for $t = 1, \ldots, G$ **do**
  $(c, s) \leftarrow \textsc{SampleCell}(\mathcal{L})$
  $a_0 \leftarrow \textsc{Elite}(\mathcal{L}, c, s)$
  $a \leftarrow \Phi_{\text{rt}}(\textsc{Mutate}(a_0, c, s))$
  $y \leftarrow g_\theta(a)$
  $v \leftarrow \Phi_{\text{safe}}(a, y)$
  if $v > \textsc{EliteScore}(\mathcal{L}, c, s)$ **then**
    $\textsc{UpdateElite}(\mathcal{L}, c, s, a, v)$
  end if
  if $v \geq \tau_{\text{atk}}$ **then**
    $n_{\text{succ}} \leftarrow n_{\text{succ}} + 1$
    $\textsc{LogForAudit}(a, y, v)$
  end if
end for
$\hat{\rho} \leftarrow 1 - \frac{n_{\text{succ}}}{G}$
return $(\mathcal{L}, \hat{\rho})$

Robustness Metrics

For attack a_t, define the success indicator

$$I_t = \mathbf{1}[\Phi_{\text{safe}}(a_t, g_\theta(a_t)) \geq \tau_{\text{atk}}]. \tag{14}$$

The attack success rate and robustness are

$$\text{ASR} = \frac{1}{G}\sum_{t=1}^{G} I_t\,, \qquad \hat{\rho} = 1 - \text{ASR}. \tag{15}$$

RIRAG reports ASR against both a frozen reference archive and the live archive. The frozen score supports comparison across models and time. The live score measures resistance to adaptive attacks discovered during the current run. For corpus poisoning, RIRAG also reports $\pi^\star$, the minimum injected-document fraction required to exceed a predefined retrieval-influence threshold [9].

Defense Feedback

Successful model attacks are added to the adversarial partition of the Golden Evaluation Dataset. Corpus and evaluator attacks become regression tests for RIRAG itself. Human audits review sampled safety judgments and prevent Φ_{safe} from becoming an unchecked point of failure. The red-team loop therefore extends the dual-closing process in Section 5.5.5: discovered attacks are converted into future tests and defensive training examples.

Operational Risk Model

Capability does not determine real-world risk by itself. Risk also depends on whether an adversary is likely to adopt the capability and, on the harm, produced by successful use [21]. RIRAG therefore combines capability, adoption, and impact into an auditable score.

Capability

Let χ denote a specific security-relevant capability, such as exploiting a class of one-day vulnerabilities or producing phishing content that evades a named filter. The Evaluation Circuit estimates the capability score $\text{Cap}(\chi, g_\theta) \in [0,1]$ from tasks that instantiate χ. Risk is defined as

$$\text{Risk}(\chi, g_\theta) = \text{Cap}(\chi, g_\theta)\, A(\chi)\, I(\chi), \tag{16}$$

where $A(\chi)$ is adversary adoption and $I(\chi)$ is impact. All factors lie in $[0,1]$. The multiplicative form assigns low risk when any required factor is near zero.

Adversary Adoption

Adoption estimates whether a rational adversary would prefer the LLM-based method to the best available baseline. We use

$$A(\chi) = \sigma(\ \beta_0 + \beta_1 \Delta c(\chi) + \beta_2 \Delta p(\chi) + \beta_3 b(\chi)), \tag{17}$$

where $\sigma(z) = 1/(1+e^{-z})$. The term $\Delta c(\chi)$ is the normalized cost advantage of the LLM method, $\Delta p(\chi)$ is its performance advantage over the baseline, and $b(\chi)$ measures how strongly it lowers the expertise barrier. The coefficients β_i are calibrated using observed behavior or structured expert elicitation. RIRAG supplies measurable inputs such as latency, token use, monetary cost, task success, and time-to-exploit.

Impact

For capabilities that affect confidentiality, integrity, or availability, impact is defined as

$$I(\chi) = \frac{E(\chi)}{Z}(w_C I_C + w_I I_I + w_A I_A), \tag{18}$$

where $I_C, I_I, I_A \in [0,1]$ are consequence scores, $E(\chi) \in [0,1]$ represents exposure or scalability, and Z

normalizes the weighted sum to $[0,1]$. The weights reflect organizational priorities. Non-CVE capabilities can use an analogous harm taxonomy supported by expert elicitation.

Aggregation and Uncertainty

A model's risk profile is the set $\{\mathrm{Risk}(\chi, g_\theta): \chi \in \mathcal{X}\}$ for capability catalogue $\mathcal{X}$. A scalar summary may be reported as

$$R_{\mathrm{all}}(g_\theta) = \sum_{\chi \in \mathcal{X}} \omega_\chi \ \mathrm{Risk}(\chi, g_\theta), \qquad \sum_{\chi} \omega_\chi = 1. \quad (19)$$

Each input is estimated with uncertainty. Capability intervals come from repeated evaluation and bootstrap resampling; adoption and impact intervals come from observed variance or expert disagreement. RIRAG propagates these distributions through Eq. (**16**) using Monte Carlo sampling and reports an interval rather than a single point estimate. The coefficients and weights remain governance choices; the framework contributes the measurable structure, not universal values.

Human-AI Teaming

Many deployed cybersecurity assistants support analysts rather than act autonomously [1,40]. A benchmark that evaluates only the model may therefore miss its actual operational value. The teaming track evaluates the analyst–model pair and measures whether assistance improves speed, accuracy, workload, and calibration.

Study Design

RIRAG uses a controlled within-subjects design. Analysts, stratified by experience, complete representative tasks such as alert triage, malware-report summarization, and CVE-to-mitigation mapping. Each participant works under at least two conditions: unaided and aided by g_θ. A conventional baseline tool may be added as a third condition. Task order and condition order are counterbalanced to reduce learning and ordering effects. Tasks are derived from STIX-grounded benchmark material so that each final decision has an expert-validated reference answer. Any study involving human participants requires institutional review and informed consent.

Teaming Metrics

Completion time. RIRAG records wall-clock time-to-resolution and tests whether assistance reduces time without reducing correctness.

Joint accuracy and uplift. Let A_{H+M}, A_H, and A_M denote team, human-only, and model-only accuracy. Complementary uplift is

$$U = A_{H+M} - \max(A_H, A_M). \quad (20)$$

Positive U indicates that the team outperforms both components individually.

Cognitive load. A validated instrument such as NASA-TLX measures mental demand, effort, frustration, and related workload dimensions after each task.

Calibration and trust. RIRAG compares analyst confidence with correctness to identify over-reliance and under-reliance. Repeated sessions can also measure how trust changes with experience [40].

Algorithm 7 Human–AI Teaming Evaluation

```
Require: participant u, tasks T
Require: conditions C, model g_θ
Ensure: time, accuracy, workload, and confidence records
T' ← COUNTERBALANCE(T, C)
for all (T, c) ∈ T' do
    STARTTIMER()
    PRESENTTASK(T, c)
    if c = model-assisted then
        ENABLEASSISTANT(g_θ)
    end if
    â_{u,T} ← COLLECTANSWER(U, T)
    l_{u,T} ← STOPTIMER()
    z_{u,T} ← SCORE(â_{u,T})
    w_{u,T} ← NASA-TLX()
    c_{u,T} ← CONFIDENCE(U)
end for
return participant records
```

Aggregate analysis uses a mixed-effects model with evaluation condition as a fixed effect and participant as a random effect. Results should report effect sizes and confidence intervals, not only p-values.

Graph Retrieval and Active Learning

Dense retrieval is effective for semantic similarity but may miss multi-hop relationships common in cyber threat intelligence. RIRAG therefore augments vector retrieval with a STIX knowledge graph and uses active learning to allocate human-review effort.

STIX Graph Construction

STIX 2.1 bundles are naturally graph structured. Vulnerabilities, attack patterns, threat actors, and indicators form nodes, while STIX relationship objects form typed edges. RIRAG merges validated packet-level bundles into a graph $\mathcal{G} = (V, E)$. Entity alignment combines exact identifiers, embedding similarity,

provenance constraints, and human review for ambiguous merges. Missing-relation prediction may be used to propose edges, but proposed relations are not treated as validated knowledge until they pass the normal review process [49,50].

Hybrid Retrieval

For query q, let $\mathcal{R}_K(q)$ be the vector-retrieved context defined in Section 5.4. Hybrid retrieval is

$$C_{\text{hyb}}(q) = \mathcal{R}_K(q) \cup \mathcal{N}_h\big(\mathcal{G}, A(q)\big), \tag{21}$$

where $A(q)$ maps the query to graph anchors and $\mathcal{N}_h$ returns their h-hop neighborhood. The vector component supplies relevant text, while the graph component supplies paths among entities and relationships. The union is deduplicated and truncated to the prompt budget. Graph consistency can also identify generated relationships that are unsupported by validated edges.

Review Prioritization

Let P be the set of packets awaiting possible review. Each packet receives the acquisition score

$$\alpha(p) = H\big(\gamma(p)\big) + \lambda_d d(p, \mathcal{K}) + \lambda_r r(p), \tag{22}$$

where H measures uncertainty, $d(p, \mathcal{K})$ measures novelty relative to the knowledge base, and $r(p)$ measures risk relevance using Section 6.2. For a confidence value $\gamma \in (0,1)$, binary entropy is

$$H(\gamma) = -\gamma \log \gamma - (1-\gamma)\log(1-\gamma). \tag{23}$$

Higher acquisition scores receive earlier review, while a small random audit sample is retained to detect systematic errors outside the prioritized set.

Algorithm 8 Active-Learning HITL Prioritization

```
Require: candidate packets P, review budget B_h
Require: weights λ_d, λ_r
Ensure: reviewed packets P_h
for all p in P do
    u_p ← H(γ(p))
    n_p ← d(p, K)
    r_p ← r(p)
    α(p) ← u_p + λ_d n_p + λ_r r_p
end for
P_h ← Top(P, α, B_h)
P_h ← P_h union AuditSample(P \ P_h)
for all p in P_h do
    p ← HumanReview(p)
    AppendTrainingSet(p)
end for
UpdateThresholds(τ-pass, τ-fail, P_h)
return P_h
```

The realized review outcomes update the triage thresholds in Eq. (**7**). This closes a second-order feedback loop: RIRAG learns both from corrected packets and from evidence about when its confidence should trigger human review.

Implementation and Case Studies

This section translates RIRAG into a reproducible system design. It describes the reference architecture, data flow, deployment constraints, and audit records needed to implement the framework. Section 7.5 then illustrates the workflow through constructed examples. These examples demonstrate system behavior but do not report measured benchmark results.

Architecture

A reference RIRAG deployment contains seven logical services connected through a durable message bus. Ingestion connectors collect data from sources such as the NVD API, MISP or STIX feeds, vendor advisories, and internal SOC documents. Each item is normalized into a Knowledge Packet containing source provenance and a SHA-256 content hash. A durable queue separates collection from downstream processing and supports replay during recovery or audit.

The Translator service hosts Φ_{tr}, while the Validator service hosts Φ_{val}. The Validator queries both a vector store and the STIX knowledge graph introduced in Section 6.4. A web-based HITL console presents the prioritized review queue and records structured corrections. The Evaluation service launches sandboxed BYOM containers, executes benchmark tasks, and hosts the Evaluator and Red-Team agents, Φ_{ev} and Φ_{rt}. A scheduler triggers retraining, evaluation, and red-team campaigns. Table **6** gives one concrete mapping from these logical components to implementation technologies.

Table 6: Reference technology mapping for a RIRAG instance.

Component	Reference implementation
Ingestion	JSON-emitting connectors with provenance and SHA-256 hashes
Message bus	Durable queue or log with replay support
Agent services	Versioned LLM endpoints with task-specific prompts
Vector store	HNSW approximate-nearest-neighbor index using ϕ
Graph store	STIX 2.1 property graph with bounded-hop queries
HITL console	Web interface for prioritized review and correction
Evaluation	Sandboxed BYOM execution, scheduling, and agent orchestration

The architecture is substrate independent. Components may be replaced provided that they preserve the packet schema and interfaces defined in Sections 5.5–5.6.

Data Flow

A packet follows the sequence

$$\begin{aligned}\text{ingest} &\rightarrow \text{queue} \rightarrow \text{translate} \\ &\rightarrow \text{validate} \rightarrow \text{review} \rightarrow \text{index/train}.\end{aligned} \tag{24}$$

Algorithm **1** formalizes this workflow.

Two consistency invariants are required. First, the *embedding invariant* requires the same embedding model ϕ for knowledge-base indexing and evaluation-time query encoding. A change to ϕ therefore requires re-indexing, and every chunk records the embedding-model version used to create it.

Second, the *gold-isolation invariant* prevents any private or held-out Golden Evaluation Dataset item from entering the training loop. Gold-derived artifacts are tagged and excluded from the aggregation performed by Algorithm **4**. Continuous-integration checks enforce both invariants because violating either one can silently invalidate evaluation results.

Deployment and Scaling

GPU inference is likely to dominate automated processing cost, while ingest-to-index delay determines how quickly the system reflects new intelligence. Translator and Validator workers are stateless and can scale horizontally behind the queue. Vector and graph stores can be partitioned by collection, tenant, or entity range.

Human review does not scale through additional compute. Its main control is the active-learning policy in Section 6.4, which directs limited expert attention toward uncertain, novel, or high-risk packets. A tiered ingestion schedule is also useful: active-exploitation alerts and high-priority vulnerability feeds can be processed continuously, while lower-priority reports are handled in batches. Retraining and large red-team campaigns can run outside peak evaluation periods.

Reproducibility

Each reported result is linked to a reconstructable set of artifacts: packet provenance and hashes, agent and embedding-model versions, the Golden Dataset snapshot identifier, the frozen red-team archive, prompt and configuration versions, per-query outputs, and bootstrap seeds. Private evaluation items need not be disclosed, but their immutable snapshot identifiers must be recorded. This makes a leaderboard result an auditable experiment rather than an isolated score and directly addresses the provenance and contamination concerns in Section 4.

Case Studies

The following constructed cases illustrate how information and decisions move through RIRAG. They are workflow examples, not empirical results.

CVE Ingestion

An ingestion connector retrieves a newly published remote-code-execution CVE affecting a widely deployed web component. The connector wraps the source record in a Knowledge Packet, records its provenance and content hash, and places it on the queue.

The Translator generates a question–answer pair, a true/false statement, a concise summary, and an IoC or TTP representation mapped to MITRE ATT&CK. Self-consistency filtering removes an unstable question–answer candidate. The Validator retrieves related vulnerabilities, creates a STIX bundle, and computes confidence using Eq. (**6**). Assume that the evidence is well grounded but that an unusual affected-version string lowers the novelty-coherence score. The packet is therefore assigned to the review band by Eq. (**7**).

The acquisition function in Eq. (**22**) prioritizes the packet because it is uncertain, novel, and security critical. A reviewer corrects the version range, verifies the STIX relationships, and approves the packet. Algorithm **4** then inserts the corrected knowledge into the vector and graph stores and appends the reviewed input–output pair to the training set. The final packet remains linked to its source and complete review history.

CTI Model Evaluation

A vendor submits a BYOM container that passes the interface and isolation checks. The framework issues a Golden Dataset query asking which ATT&CK techniques are associated with a specified threat actor and which mitigations apply. It retrieves vector context with $\mathcal{R}_K$ and expands the relevant graph neighborhood using Eq. (21).

The prompt contains the query, the retrieved prose, and graph-derived relationships. The submitted model returns an answer and source references. The

Evaluator Agent decomposes the answer into claims and computes faithfulness, answer relevance, and factual correctness using Eqs. (9)–(12). A claimed actor–technique relationship that is absent from both the retrieved text and graph is marked unsupported. Per-claim scores are retained for diagnosis and later aggregated across repeated runs with confidence intervals.

Retrieval Poisoning

The Red-Team Agent generates documents that promote a false mitigation and optimizes them for similarity to likely queries, following the threat demonstrated by PoisonedRAG [9]. The documents enter through a simulated untrusted feed.

During validation, the proposed mitigation conflicts with trusted knowledge. The consistency score γ_c falls, and the geometric aggregation in Eq. (6) drives the overall confidence below the acceptance threshold. The packets are routed to human review rather than indexed. The framework records the attack, its poisoning budget $\pi^{\star}$, and the originating source. If no trusted contradiction were available, audit sampling and human review would provide the remaining detection boundary.

Risk Scoring

The operator evaluates the capability χ defined as autonomously exploiting a specified class of one-day vulnerabilities from public descriptions. The Evaluation Circuit estimates $\mathrm{Cap}(\chi, g_\theta)$ from repeated task outcomes. Eq. (**17**) estimates adoption using measured cost, time-to-exploit, performance uplift over a baseline, and reduction in required expertise. Eq. (**18**) estimates impact from confidentiality, integrity, availability, and exposure.

The framework combines these factors using Eq. (**16**) and propagates their uncertainty through Monte Carlo sampling. This procedure can distinguish models with similar task success but different costs, accessibility, or likely consequences. It therefore reports operational risk rather than capability alone.

Failure Diagnostics

Section 4 identified a basic limitation of aggregate scores: they show that a model failed, but rarely explain why. RIRAG addresses this problem through a structured execution log. For each evaluated query, the log records the retrieved context, generated answer, metric vector, evaluator decision, and an attributed failure category. The resulting record turns a leaderboard score into a diagnostic signal.

Failure Taxonomy

RIRAG assigns unsuccessful outputs to one of six categories. The taxonomy reflects the main stages of a RAG pipeline—retrieval, knowledge coverage, reasoning, instruction following, grounded generation, and tool execution. Human auditors review a sample of the assigned labels and may revise them when the automated attribution is uncertain. Table **7** summarizes the categories and their operational implications.

Table 7: Failure modes recorded in the RIRAG execution log.

Failure mode	Detection signal	Corrective action
Retrieval miss	Required gold context is present in the knowledge base but absent from the retrieved set; context recall is low (Eq. (11)).	Revise the embedding or index; adjust retrieval depth or graph expansion.
Knowledge gap	Required evidence is absent from both the retrieved set and the current knowledge base.	Acquire and validate the missing knowledge through the Knowledge Circuit.
Reasoning error	Relevant evidence is retrieved, but the conclusion is unsupported or logically inconsistent.	Use targeted reasoning data and audit the structured execution trace.
Prompt misinterpretation	The response addresses the wrong task or sub-question; answer relevance is low (Eq. (10)).	Clarify task instructions and improve instruction tuning.
Hallucination	One or more claims are unsupported by the supplied context; faithfulness is low (Eq. (9)).	Strengthen grounding constraints and penalize unsupported claims.
Tool or format error	The model violates the output schema, produces invalid arguments, or fails to complete a required tool call.	Repair the interface contract and add tool-use training examples.

Diagnostics Use

The Evaluator Agent performs the initial attribution, and human reviewers audit a sample of its labels. Each label is stored with the raw metrics and execution artifacts. Aggregating these records over the Golden Evaluation Dataset produces a *failure profile*: the proportion of failures assigned to each category.

This profile distinguishes models that appear similar under aggregate accuracy. One model may fail mainly because relevant evidence was not retrieved, indicating an integration problem. Another may receive the correct evidence but draw the wrong conclusion, indicating a model-capability problem. The required interventions are different.

The same profile also guides RIRAG's internal feedback loops. A growing share of knowledge-gap failures suggests that ingestion is not keeping pace with the threat landscape. A rise in reasoning errors suggests that retraining should emphasize multi-step, evidence-based tasks. Diagnostic logging therefore supplies a control signal to the dual-closing loop in Section 5.5.5, rather than merely producing a post-hoc explanation.

Diagnostic Example

Consider the CTI query in Case Study 2 (Section 7.5). Suppose a submitted model recommends a plausible but incorrect mitigation. The correct evidence was included in the retrieved subgraph, so context recall remains high. The mitigation claim is nevertheless unsupported by that evidence, producing a faithfulness violation. RIRAG therefore records a hallucination rather than a retrieval miss.

The distinction matters operationally. Expanding the knowledge base would not correct this failure because the model already received the necessary evidence. A deployment team should instead place greater weight on faithfulness and examine whether the model follows grounding instructions under high-stakes conditions. A single accuracy score would not expose this difference.

Deployment and the Attack–Defense Landscape

Deployment Scenarios

RIRAG exposes several configurable operating points: the triage thresholds τ_{pass} and τ_{fail} from Eq. (**7**), the human-review budget B_h, the metric weights λ_k from Eq. (**13**), and the red-team budget G. Their settings should reflect the deployment context rather than a universal default.

Managed Cloud SOC

A managed detection-and-response provider may use RIRAG to compare assistants used by security analysts. Such a deployment places strong emphasis on liveness and faithfulness because intelligence changes quickly and unsupported recommendations can affect active investigations. The operator may therefore select a high pass threshold, allocate substantial expert-review capacity, assign greater weight to faithfulness, and run frequent adversarial evaluations. The human–AI teaming track in Section 6.3 is especially relevant because the deployed unit is the analyst–model team rather than the model alone [40].

Air-Gapped Infrastructure

An industrial-control operator may receive intelligence through periodic, vetted transfers rather than continuous external feeds [37]. In this setting, update frequency is lower, but provenance and robustness requirements are stricter. Every imported item should be traceable, and the offline corpus should be tested for poisoning before it is admitted.

The operator may also assign greater weight to availability impact I_A when computing $\text{Impact}(\chi)$ in Eq. (**18**). A larger human-review fraction and more aggressive corpus-poisoning tests may be justified when the cost of an incorrect recommendation is high.

Benchmark-as-a-Service

A neutral organization operating RIRAG as a public benchmark prioritizes standardization, reproducibility, and contamination resistance. It can freeze a public reference set and red-team archive for longitudinal comparison while rotating a private freshness partition to limit memorization. The embedding model, evaluator, prompt templates, and software versions should be fixed for each benchmark release.

Results should include uncertainty intervals and per-category failure profiles, not only a composite rank. Risk-model coefficients should also be disclosed when risk scores are published. These choices make the operating point visible and allow users to judge whether the benchmark matches their intended deployment.

Attack–Defense Landscape

RIRAG combines retrieval, generation, automated judging, model updates, and external data ingestion. It therefore inherits the attack surfaces of each component. Table **8** maps the main attack classes to the framework mechanisms intended to reduce their effect. The mapping supports a defense-in-depth position: no single safeguard is sufficient, and evaluation infrastructure must itself be treated as part of the security boundary [51–53].

The listed controls reduce risk but do not establish immunity. Their effectiveness must be measured under adaptive attacks and reviewed as the framework, models, and threat landscape change.

Table 8: Attack classes and corresponding RIRAG mitigations.

Attack class	Representative work	RIRAG mitigation
Direct jailbreak	[7,25]	Adaptive red-team loop and hardened evaluation prompts.
Indirect prompt injection	[6,8]	Delimited context, instruction isolation, and judge-injection tests.
Retrieval poisoning	[9,27]	Provenance tracking, consistency scoring, quarantine, and HITL review.
Adaptive evasion	[26]	A continuously updated adversarial archive with frozen and live evaluations.
Fine-tuning poisoning	[28]	Gold-set validation, deployment gates, and rollback of regressive updates.
Autonomous exploitation	[14,15,54]	Controlled capability evaluation and risk scoring without releasing operational attack procedures.

Empirical Validation Protocol

This section defines a preregistrable, falsifiable protocol for testing the claims made by RIRAG. It is the empirical companion to the analytical contributions of this article: executing it is the scoped future work by which the framework stands or falls. It does not substitute planned measurements with illustrative numbers. Instead, it states falsifiable hypotheses, required baselines, and confirmatory analyses so that a later implementation can be evaluated reproducibly.

Hypotheses

H1 Liveness. Performance on a private freshness partition containing recently disclosed threats will be lower than performance on an older partition, and the difference will be underestimated by a contaminated static benchmark.

H2 Diagnostic value. Interventions matched to an attributed failure category will reduce that category more than unmatched interventions. Adding missing knowledge, for example, should reduce knowledge-gap failures more than reasoning errors.

H3 Self-improvement. Under a stable input distribution, the human-review rate h_t will decline across update epochs without reducing performance on the held-out gold set.

H4 Adaptive robustness. For the same model, attack success rate against the live archive will exceed attack success rate against the frozen reference archive.

H5 Risk discrimination. Risk-based rankings will differ from capability-only rankings when models have materially different adoption costs or when the evaluated capabilities have different impacts.

H6 Teaming effect. For pre-specified SOC tasks, model assistance will change analyst accuracy, completion time, or cognitive load relative to the unaided condition. The direction and size of each effect will be reported rather than assumed.

Datasets, Baselines, and Metrics

The Golden Evaluation Dataset is divided into a public, frozen comparability partition and a private, rotating freshness partition. The study should include proprietary and open models spanning a meaningful capability range. A retrieval-only baseline isolates retrieval performance, while a closed-book baseline measures behavior without access to the curated knowledge base. Where practical, a conventional non-LLM security tool should be included as an operational baseline.

Primary outcomes include the retrieval and generation metrics in Table **5**, attack success rate from Eq. (**15**), risk from Eq. (**16**), review rate h_t, and the teaming measures defined in Section 6.3. Metrics should be reported with bootstrap confidence intervals over queries and repeated runs.

Table 9: Validation roadmap for the RIRAG hypotheses.

Hyp	Manipulation or measurement	Confirmatory analysis
H1	Compare recent and aged partitions, then repeat on the static set.	Paired estimate of the freshness gap and its confidence interval.
H2	Apply knowledge and reasoning interventions to categorized failures.	Test the intervention-by-failure-category interaction.
H3	Run the dual-closing loop across several update epochs.	Estimate the trend in h_t subject to a non-inferiority guard on gold performance.
H4	Evaluate each model against frozen and live attack archives.	Paired within-model difference in attack success rate.
H5	Compute capability and risk rankings for the same catalogue.	Rank correlation with analysis of materially divergent cases.
H6	Conduct a counterbalanced, within-subjects teaming study.	Mixed-effects analysis of accuracy, time, and NASA-TLX outcomes.

Validity Controls

The analysis plan should be preregistered to limit post-hoc selection. The teaming study should counterbalance task order and assistance condition. Model, evaluator, prompt, embedding, and index versions must be pinned for each run, and the gold-isolation invariant from Section 7 must prevent evaluation items from entering the training loop.

Provider-hosted models may change without retaining the same public name. Each result should therefore record the provider, exact model identifier, access date, decoding parameters, and relevant endpoint version. Repeated runs and uncertainty intervals do not remove model stochasticity, but they prevent a single realization from being reported as a stable ranking.

Related Work and Positioning

RIRAG draws from several research areas: foundation-model evaluation, retrieval-augmented generation, agentic systems, adversarial robustness, machine learning for security, threat-intelligence processing, and human–AI collaboration. This section positions the framework within those areas and distinguishes the components it adopts from the mechanisms it combines or extends.

Foundation Models

Transformers [55] and large-scale pre-training [56–60] established the language and reasoning capabilities now applied to cybersecurity. Instruction tuning and preference-based optimization further improved task following and behavioral alignment [61–68]. Open-weight LLaMA models [69,70], large proprietary families such as PaLM [71], and parameter-efficient methods including LoRA and QLoRA [72,73] made domain adaptation increasingly practical.

Scaling studies [74,75] and reports of emergent behavior [76,77] also exposed a central evaluation problem: capability cannot be inferred reliably from model size or training loss and must be measured directly [78,79]. General benchmarks such as GLUE, SuperGLUE, MMLU, BIG-bench, AGIEval, and MMLU-Pro [17,18,80–83], together with reasoning, coding, and truthfulness suites [84–90], provide useful evaluation patterns but also illustrate saturation and contamination risks. Domain benchmarks in law and medicine [91,92] support the case for similarly specialized cybersecurity evaluation.

RIRAG's Evaluator Agent follows work on LLM-based judging, including MT-Bench and Chatbot Arena [93], while its claim-level checks are related to reference-free factuality methods such as FActScore and SelfCheckGPT [94,95]. RIRAG differs by treating the judge as part of the attack surface and subjecting its decisions to audit and adversarial testing.

RAG and Security

RIRAG builds on retrieval-augmented generation [96–99], dense and late-interaction retrieval [100,101], and approximate-nearest-neighbor indexing [102,103]. Reflective and retrieval-controlled variants [104,105], RAG surveys [23,106], and RAGAS [24] inform the framework's iterative workflow and evaluation metrics.

Retrieval also creates security risks. Corpus poisoning and knowledge-corruption attacks can steer generation without modifying model parameters [9,27,107,108], while certified or filtering-based defenses seek to bound that influence [109]. RIRAG therefore treats ingestion, validation, and indexing as security-sensitive operations rather than neutral preprocessing steps. Provenance records, confidence-based triage, human review, and corpus-focused red teaming are used together because no single control eliminates the poisoning surface.

LLM Agents in Cybersecurity

Reasoning-and-action loops, tool use, reflection, and multi-agent coordination provide the technical basis for agentic systems [110–117]. General agent benchmarks such as Agent Bench [118] measure this broader capability, while cybersecurity studies examine website exploitation [119], one-day and zero-day vulnerability exploitation [14,15], guided attack execution [120], penetration testing [16,121], and coordinated vulnerability discovery [54].

Prompt-injection benchmarks for tool-using agents, including InjecAgent and AgentDojo [122,123], are especially relevant to RIRAG because submitted models, retrieved documents, and evaluation prompts can all carry adversarial instructions. Surveys of agentic cyber capability [13,124] further motivate the framework's distinction between demonstrated capability and operational risk.

Jailbreaks and Red Teaming

A large body of work shows that aligned models remain vulnerable to jailbreaks, optimization-based attacks, role-play, obfuscation, and adaptive search [7,125–133]. Prompt injection extends this threat to

applications that combine models with external data and tools [6,8,134–137]. Work on extraction, memorization, and adversarial triggers shows that risk can also originate in training data and learned representations [138–140]. Adaptive attacks are particularly important because they can invalidate conclusions drawn from fixed defenses [26].

Automated red-teaming methods [48,53,141–144] and standardized suites such as Jailbreak Bench and Harm Bench [25,145,146] inform RIRAG's evolving attack archive. Defensive methods, including input transformations, randomized smoothing, certification, and safety classifiers [147–150], inform its mitigation and regression-testing layers. Harm taxonomies [151] support attack categorization, while hallucination surveys [152,153] motivate the framework's faithfulness and factuality checks.

Machine Learning for Security

RIRAG also inherits methodological lessons from machine learning for security. Early work warned that closed-world evaluation often fails to represent deployment conditions [154]; later guidance documented recurring errors in dataset construction, experimental design, and interpretation [155,156]. These concerns motivate RIRAG's emphasis on provenance, fresh data, held-out evaluation, uncertainty reporting, and deployment-specific operating points.

Intrusion-detection and malware research provides both useful data and cautionary examples. Widely used datasets [157–160], malware models [161,162], and surveys [163,164] demonstrate the value of learned security systems but also their sensitivity to distribution shift and adversarial preprocessing [165]. Phishing detection [166] further illustrates how automation can improve defense while simultaneously lowering the cost of scalable attacks, a trade-off captured by RIRAG's adoption and impact factors.

Code Security and Threat Intelligence

RIRAG's applied evaluation partitions draw on neural vulnerability detection [167–171], code-focused pre-trained models [172–177], and studies of LLM-assisted secure coding and repair [178–182]. This literature supplies task formats and known failure modes, including insecure generation, weak localization, and sensitivity to superficial code changes.

The knowledge-graph extension builds on automated threat-intelligence extraction, attack-graph construction, and provenance reasoning [49,50,183–188]. MITRE ATT&CK [189] provides the shared vocabulary used to connect threat actors, techniques, indicators, and vulnerabilities within the STIX-based graph.

Human-AI Security Operations

The teaming track is informed by research on complementary human–AI performance [46], overreliance and cognitive forcing [47], and broader empirical reviews of human–AI decision making [190]. Security-operations studies document alert fatigue [191,192], SOC organization [193], and emerging patterns of LLM use by analysts [40]. These findings support evaluating analyst–model teams rather than assuming that standalone model accuracy predicts operational benefit.

Recent cybersecurity-focused LLM surveys [51,52,194–200] describe rapidly expanding offensive, defensive, and evaluation applications. Within the literature surveyed in this article, these elements are usually studied separately. RIRAG's contribution is

Table 10: Positioning of RIRAG relative to representative evaluation frameworks. A check mark denotes explicit support, ~ partial support, and an em dash no primary support.

Framework	Living	Diagnostic	Risk	Teaming	Adversarial	Self-update
CSEBenchmark [4]	—	✓	—	—	—	—
VulDetectBench [33]	—	~	—	—	—	—
SecLLMHolmes [20]	—	~	—	—	✓	—
NYU CTF / Cybench [35,39]	—	~	~	—	—	—
PentestGPT [16]	—	—	—	~	—	—
CTIBench / CTIArena [5,36]	—	~	—	—	—	—
CyberSOCEval [1]	~	~	—	~	—	—
NetSPI [38]	—	—	~	—	✓	—
Risk-aware critique [21]	—	—	✓	—	—	—
RIRAG	✓	✓	✓	✓	✓	✓

their integration into one auditable workflow that combines continuously updated knowledge, diagnostic evaluation, adversarial self-testing, human review, risk estimation, and human–AI teaming analysis.

Positioning RIRAG

RIRAG incorporates tasks and insights from existing cybersecurity benchmarks rather than replacing them. Its main contribution is to place those evaluations inside a living, risk-aware, and adversarially tested workflow. Table **10** summarizes this relationship across the dimensions identified in Section 4.

Existing Benchmarks

Knowledge-oriented benchmarks provide reusable task formats and taxonomies. CSEBenchmark, CyberMetric, SecBench, and CS-Eval can inform the knowledge partition of the Golden Evaluation Dataset [4,30–32]. Applied benchmarks provide templates for specialized partitions. VulDetectBench contributes progressively difficult code-analysis tasks, while SecLLMHolmes contributes robustness-oriented transformations [20,33]. NYU CTF Bench and Cybench provide interactive tasks whose success measures can inform the capability term in the risk model [35,39]. CTIBench and CTIArena contribute CTI tasks that align naturally with RIRAG's STIX-structured knowledge [5,36].

RIRAG hosts these task families within a common pipeline that controls freshness, records uncertainty, separates the evaluator from the model under test, and connects capability measurements to operational risk.

Distinctive Contributions

Within the frameworks surveyed here, RIRAG differs in three respects. First, it closes the evaluation loop. Validated knowledge updates the retrieval substrate, while reviewed corrections can improve the framework's internal agents. Second, it operationalizes risk by combining capability, adversary adoption, and impact with explicit uncertainty, extending the conceptual argument in [21]. Third, it treats the benchmark infrastructure as part of the attack surface. The Red-Team Agent tests not only submitted models but also the retrieval corpus and evaluation path, building on evidence from robustness, poisoning, and adaptive-attack research [9,20,26,38].

Broader Context

RIRAG connects several active research directions. Surveys organize attacks, defenses, and security controls for LLM systems [51,52]. Other work studies autonomous exploitation and agentic offensive capability [13–15,54,124,201]. Automated red-teaming has produced adaptive and quality–diversity methods [25,48,53], while RAG evaluation research has developed metric suites and evaluation taxonomies [23,24]. Cybersecurity fine-tuning studies also expose tensions between task performance and safety [28].

RIRAG integrates these strands into one evaluation architecture: RAG metrics assess grounded generation, red-team search probes adaptive failure, STIX data support structured CTI reasoning, and the risk model connects measured capability to likely operational consequence.

DISCUSSION AND FUTURE WORK

Limitations

The most important limitation is scope: this article specifies and analyzes the framework but does not implement or empirically validate it. All claims about RIRAG's effectiveness are therefore design arguments derivations from documented failure modes (Section 4.5) and established evaluation theory (Section 5.1) rather than measured results. Section 9 converts those arguments into falsifiable hypotheses with named datasets, baselines, and analyses precisely so that this gap can be closed by future experimental work.

RIRAG inherits the weaknesses of its components. The Translator, Validator, Evaluator, and Red-Team agents are all fallible. Self-consistency, confidence scoring, held-out validation, and human review reduce error but cannot remove it, and each safeguard adds computational or labor cost.

The Evaluator Agent is independent of the model under test, but independence does not make it objective. It may contain systematic bias, miss subtle factual errors, or be manipulated by candidate content. Hardened prompts and periodic human audits therefore provide risk reduction rather than proof of correctness.

The risk model also depends on estimated adoption coefficients and impact weights. Reporting uncertainty makes those assumptions visible, but it does not eliminate subjectivity. Human review presents a similar trade-off: it is the framework's main trust anchor, yet also its largest scaling constraint and a possible source of inconsistent judgments. Active learning concentrates expert effort where it is most useful, but expert labor remains necessary.

Threats to Validity

Construct validity. Faithfulness, robustness, and risk are represented through measurable proxies. These proxies may omit relevant behavior or become targets for optimization. RIRAG addresses this concern through multiple metrics, diagnostic logging, adversarial evaluation, and human audit, but construct validity must still be tested empirically.

Internal validity. The framework contains several interacting components. A change attributed to retrieval, retraining, or graph augmentation may instead arise from another part of the pipeline. The modular interfaces and execution logs support controlled ablations, but they do not replace them.

External validity. The worked case studies demonstrate workflow and interfaces, not predictive validity in operational deployments. Whether RIRAG scores correspond to incident reduction, analyst performance, or attacker adoption remains an open empirical question.

Reproducibility. Stochastic generation prevents exact reruns even when prompts and model versions are fixed. RIRAG records versions, seeds, raw outputs, and score distributions, and reports confidence intervals rather than single values. Residual variability remains and should be treated as part of the result.

Ethical and Dual Use

RIRAG is itself dual-use. The Red-Team Agent generates adversarial inputs, and the risk model identifies capabilities with high misuse potential. These functions can strengthen defenses, but they may also expose operational attack patterns or lower the cost of misuse if released without safeguards.

Responsible operation therefore requires access controls for adversarial archives, review of human-subjects studies, governance for risk-model inputs, and restraint when publishing exploit details. High-risk artifacts should be shared according to demonstrated defensive need rather than by default. The framework should also record who can access attack data and how those data are used.

Cybersecurity fine-tuning can introduce an additional safety–performance trade-off [28]. Training on offensive security data may improve domain performance while weakening broader safety behavior. RIRAG's held-out validation and rollback gate are intended to detect such regressions, but they cannot guarantee that every harmful behavioral change will be observed. Ethical controls are therefore part of the evaluation design, not an external administrative layer.

Future Directions

The framework and analysis above suggest a focused research agenda. The priorities below follow the same dimensions used in our gap analysis, but recast them as problems that can be tested empirically.

Living Benchmarks

The Knowledge Circuit makes continuous updating technically possible, but several questions remain unresolved. The first is freshness: how quickly must new intelligence move from ingestion to evaluation for a benchmark to test reasoning about an emerging threat before that threat becomes widely documented? The second is comparability. A benchmark that changes over time must preserve stable reference partitions while maintaining a rolling live set, otherwise scores from different dates cannot be interpreted together. The third is contamination. Private items may eventually enter training corpora, so benchmark operators need methods for detecting and accounting for prior exposure.

Promising directions include contamination canaries, time-stratified evaluation relative to a model's training cutoff, and cryptographic commitments to hidden gold sets. Such commitments would permit later verification of an evaluation snapshot without disclosing its contents in advance.

Risk-Model Validation

The model in Section 6.2 defines a computable risk structure, but its adoption coefficients and impact estimates require empirical calibration. Future studies should compare predicted adoption with observed attacker behavior, including when threat actors select LLM-based methods instead of established tools. Impact mappings should likewise be tested against incident data and organizational loss models.

Surveys of agentic offensive capability [13,124] provide an initial evidence base, but reliable calibration will require collaboration with organizations that observe real misuse. Governance is inseparable from this process. Institutions must document who sets capability priorities, how elicited inputs are reviewed, and how risk scores influence procurement, access control, or deployment decisions.

Teaming Evaluation

The teaming track in Section 6.3 should be tested with larger and more diverse analyst populations. Longitudinal studies are especially important because trust, reliance, and working practices change as analysts gain experience with a model [40]. Open problems include designing tasks that remain operationally realistic while still being reproducible, controlling model-version changes during long studies, and defining metrics that transfer across SOCs with different staffing and maturity levels.

A further measurement question concerns the meaning of uplift. Short-term task improvement may not capture learning, deskilling, or overreliance that develops over months. Future evaluations should therefore distinguish immediate assistance from longer-term changes in analyst capability.

Robustness Standards

Section 6.1 treats robustness as a continuously measured property. Comparable reporting, however, requires shared adversarial archives and common conventions for frozen-archive ASR, live-archive ASR, poisoning budgets, and judge-injection rates. These artifacts should be versioned and accompanied by transfer tests against previously unseen attacks.

The central lesson of adaptive-attack research is that defenses must be evaluated against an attacker who observes and responds to them [26]. This should become a routine expectation for cybersecurity benchmarks rather than an optional stress test.

Academia-Industry Collaboration

Operationally realistic benchmarks depend on data about attacker behavior, defender workflows, and deployment constraints that are often held by industry. Academic groups, by contrast, usually provide the methodological controls needed for valid evaluation [21]. Shared-task competitions, research consortia, and privacy-preserving data-access agreements could help bridge this divide. RIRAG's provenance and audit mechanisms provide one possible foundation for controlled, accountable sharing of operational evidence.

CONCLUSION

This article examined the development of LLM cybersecurity benchmarks from knowledge-oriented tests to applied evaluations of vulnerability analysis, offensive operations, and threat intelligence, and then to emerging frameworks concerned with risk and usability. Across these settings, a consistent limitation appears: models often demonstrate broad factual and conceptual knowledge while remaining weaker on procedural execution, causal reasoning, semantic code analysis, and adversarial robustness.

We identified five gaps in current evaluation practice: the separation between capability and operational risk, limited failure diagnostics, weak coverage of human–AI teaming, staticity and contamination, and insufficient testing of the benchmark infrastructure itself. These gaps make it difficult to determine not only whether a model can complete a task, but whether its behavior is dependable, useful, and safe in practice.

RIRAG addresses these problems through two coupled feedback systems. The Knowledge Circuit ingests, validates, and indexes current cybersecurity intelligence while producing reviewed data for agent improvement. The Evaluation Circuit tests submitted models with held-out tasks, retrieval- and generation-specific metrics, diagnostic logging, uncertainty estimates, and an independent evaluator. Four extensions add continuous adversarial testing, an operational risk model, a human–AI teaming track, and hybrid graph retrieval with active-learning review. The implementation blueprint and worked cases specify how these components can be built and evaluated without presenting unmeasured performance claims as results.

The framework remains a research design rather than a validated production system. Its central proposal is therefore methodological: cybersecurity evaluation should move beyond the question of whether a model can perform an isolated task. It should also measure how current the evidence is, why failures occur, how an adaptive attacker changes the result, what risk follows from successful use, and whether the model improves the performance of a human team. Establishing those properties is necessary before LLM-based security systems can be treated as trustworthy operational partners.

AUTHOR'S CONTRIBUTION

The authors are from the industry and had helped us validate the concepts and architectures used in the paper. The authors also reviewed the version you sent and have suggested the edits.

CONFLICT OF INTEREST

There is no conflict with these authors.

APPENDIX A: NOTATION

Table **11** summarizes the notation used in Sections 5–6.4.

Table 11: Summary of notation.

Symbol	Meaning
$\mathcal{K} = \{c_i\}_{i=1}^{N}$	Indexed knowledge base containing N chunks.
$\phi,\ \mathbf{e}_i$	Fixed embedding model and chunk embedding, where $\mathbf{e}_i = \phi(c_i)$.
p	Knowledge Packet $\langle \mathrm{id}, m, r, T, S, v \rangle$.
$\Phi_{\mathrm{tr}},\ \Phi_{\mathrm{val}},\ \Phi_{\mathrm{ev}}$	Translator, Validator, and Evaluator agents.
$\Phi_{\mathrm{rt}},\ \Phi_{\mathrm{safe}}$	Red-Team Agent and safety judge.
$\gamma,\ \gamma_g,\ \gamma_c,\ \gamma_n$	Composite, groundedness, consistency, and novelty scores.
$\tau_{\mathrm{pass}},\ \tau_{\mathrm{fail}},\ \tau_{\mathrm{atk}}$	Triage and attack-success thresholds.
$\mathcal{D}_g$	Golden Evaluation Dataset $\{(q_j, a_j^\star, C_j^\star)\}_{j=1}^{M}$.
g_θ	Model under test.
$\mathcal{R}_K(q)$	Operator returning the K highest-scoring knowledge chunks for query q.
$\mathrm{Cap},\ \mathrm{Adopt},\ \mathrm{Impact},\ \mathrm{Risk}$	Capability, adoption, impact, and risk functions.
$\mathcal{G},\ \mathcal{L}$	STIX knowledge graph and red-team attack archive.
$\alpha(p)$	Active-learning acquisition score for packet p.

APPENDIX B: GLOSSARY OF ACRONYMS

Table 12: Acronyms used in this article.

Acronym	Expansion
ANN	Approximate Nearest Neighbor
ASR	Attack Success Rate
ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
BYOM	Bring Your Own Model
CTI	Cyber Threat Intelligence
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
HITL	Human-in-the-Loop
HNSW	Hierarchical Navigable Small World
ICS	Industrial Control System
IoC	Indicator of Compromise
LLM	Large Language Model
MCQ	Multiple-Choice Question
NVD	National Vulnerability Database
QA	Question–Answer
RAG	Retrieval-Augmented Generation
RIRAG	Reflective and Iterative Retrieval-Augmented Generation
SOC	Security Operations Center
STIX	Structured Threat Information Expression
TTP	Tactics, Techniques, and Procedures

REFERENCES

[1] Deason L, Bali A, Bejean C, Bolocan D, Crnkovich J, Croitoru I, *et al.* CyberSOCEval: Benchmarking LLMs Capabilities for Malware Analysis and Threat Intelligence Reasoning. arXiv preprint arXiv:2509.20166; 2025.

[2] Microsoft Security. Microsoft Copilot for Security. 2024. Available from: https://www.microsoft.com/enus/security/business/aimachine-learning/microsoft-copilot-security.

[3] Google Cloud. Supercharge security with AI. 2024. Available from: https://cloud.google.com/security/ai.

[4] Wang D, Zhou G, Li X, Bai Y, Chen L, Qin T, *et al.* The Digital Cybersecurity Expert: How Far Have We Come? arXiv preprint arXiv:2504.11783; 2025.

[5] Alam MT, Bhusal D, Nguyen L, Rastogi N. CTIBench: A benchmark for evaluating LLMs in cyber threat intelligence. arXiv preprint arXiv:2406.07599; 2024.

[6] Perez F, Ribeiro I. PromptInject: A framework for quantitative analysis of robustness to adversarial prompt attacks. arXiv preprint arXiv:2211.09527; 2022.

[7] Zou A, Phute Z, Bubeck S, Ribeiro MT. Universal and transferable adversarial attacks on aligned language models. arXiv preprint arXiv:2307.15043; 2023.

[8] Liu Y, Jia Y, Jia J, Song D, Gong NZ. DataSentinel: A Game-Theoretic Detection of Prompt Injection Attacks. In: Proceedings of the 2025 IEEE Symposium on Security and Privacy (SP); 2025.

[9] Zou W, Geng R, Wang B, Jia J. PoisonedRAG: Knowledge Corruption Attacks to Retrieval-Augmented Generation of Large Language Models. In: Proceedings of the 34th USENIX Security Symposium (USENIX Security 25); 2025.

[10] Srikanth S, Hasanuzzaman M, Meem FT. Evaluating the usability of LLMs in threat intelligence enrichment. arXiv preprint arXiv:2409.15072; 2024.

[11] Liu Z, Shi J, Buford J. CyberBench: A multi-task benchmark for evaluating large language models in cybersecurity. In: Proceedings of the AAAI Conference on Artificial Intelligence; 2024.

[12] Ji H, Yang J, Chai L, Wei C, Yang L, Duan Y, *et al.* SevenLLM: Benchmarking, eliciting, and enhancing abilities of large language models in cyber threat intelligence. arXiv preprint arXiv:2405.03446; 2024.

[13] He M, *et al.* Forewarned is Forearmed: A Survey on Large Language Model-based Agents in Autonomous Cyberattacks. arXiv preprint arXiv:2505.12786; 2025.

[14] Fang R, Bindu R, Gupta A, Kang D. LLM Agents can Autonomously Exploit One-day Vulnerabilities. arXiv preprint arXiv:2404.08144; 2024.

[15] Fang R, Bindu R, Gupta A, Kang D. Teams of LLM Agents can Exploit Zero-Day Vulnerabilities. In: Proc. 18th Conf. European Chapter of the ACL (EACL); 2026.

[16] Deng G, Liu Y, Mayoral-Vilches V, Liu P, Li Y, Xu Y, *et al.* PentestGPT: Evaluating and Harnessing Large Language Models for Automated Penetration Testing. arXiv preprint arXiv:2308.06782; 2024.

[17] Wang A, Singh A, Michael J, Hill F, Levy O, Bowman SR. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In: Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP; 2018. p. 353–355. doi:10.18653/v1/W18-5446.

[18] Hendrycks D, Burns C, Basart S, Zou A, Mazeika M, Song D, *et al.* Measuring massive multitask language understanding. arXiv preprint arXiv:2009.03300; 2020.

[19] Liang P, Bommasani R, Lee T, Tsipras D, Soylu D, Yasunaga M, *et al.* Holistic evaluation of language models. arXiv preprint arXiv:2211.09110; 2022.

[20] Ullah S, Han M, Pujar S, Pearce H, Coskun A, Stringhini G. LLMs Cannot Reliably Identify and Reason About Security Vulnerabilities (Yet?): A Comprehensive Evaluation, Framework, and Benchmarks. arXiv preprint arXiv:2312.12575; 2024.

[21] Lukošiūtė K, Swanda A. LLM Cyber Evaluations Don't Capture Real-World Risk. arXiv preprint arXiv:2502.00072; 2025.

[22] Alrashedy K, Aljasser A. Can LLMs patch security issues? arXiv preprint arXiv:2312.00024; 2023.

[23] Yu A, *et al.* Retrieval Augmented Generation Evaluation in the Era of Large Language Models: A Comprehensive Survey. arXiv preprint arXiv:2504.14891; 2025.

[24] Es S, James J, Espinosa-Anke L, Schockaert S. RAGAS: Automated Evaluation of Retrieval Augmented Generation. In: Proc. 18th Conf. European Chapter of the ACL: System Demonstrations; 2024. p. 150–158. doi:10.18653/v1/2024.eacl-demo.16.

[25] Chao P, Debenedetti E, Robey A, Andriushchenko M, Croce F, Sehwag V, *et al.* JailbreakBench: An Open Robustness Benchmark for Jailbreaking Large Language Models. In: Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track; 2024.

[26] Nasr M, Carlini N, *et al.* The Attacker Moves Second: Stronger Adaptive Attacks Bypass Defenses Against LLM Jailbreaks and Prompt Injections. arXiv preprint arXiv:2510.09023; 2025.

[27] Cho S, *et al.* Towards More Robust Retrieval-Augmented Generation: Evaluating RAG Under Adversarial Poisoning Attacks. arXiv preprint arXiv:2412.16708; 2024.

[28] ElZemity A, Arief B, Li S. CyberLLMInstruct: A Pseudo-malicious Dataset Revealing Safety-performance Trade-offs in Cyber Security LLM Fine-tuning. arXiv preprint arXiv:2503.09334; 2025.

[29] Li G, Li Y, Wang G, Yang H, Yu Y. SecEval: A comprehensive benchmark for evaluating cybersecurity knowledge of foundation models. GitHub; 2023. Available from: https://github.com/XuanwuAI/SecEval.

[30] Tihanyi N, Ferrag MA, Jain R, Bisztray T, Debbah M. CyberMetric: A benchmark dataset based on retrieval-augmented generation for evaluating LLMs in cybersecurity knowledge. arXiv preprint arXiv:2402.07688; 2024.

[31] Jing P, *et al.* SecBench: A Comprehensive Multi-Dimensional Benchmarking Dataset for LLMs in Cybersecurity. arXiv preprint arXiv:2412.20787; 2024.

[32] Yu Z, Zeng J, Chen S, Xu W, Xu D, Liu X, *et al.* CS-Eval: A Comprehensive Large Language Model Benchmark for CyberSecurity. arXiv preprint arXiv:2411.16239; 2024.

[33] Liu Y, Gao L, Yang M, Xie Y, Chen P, Zhang X, *et al.* VulDetectBench: Evaluating the deep capability of vulnerability detection with large language models. arXiv preprint arXiv:2406.07595; 2024.

[34] Bhatt M, Chennabasappa S, Nikolaidis C, Wan S, Evtimov I, Gabi D, *et al.* Purple Llama CyberSecEval: A secure coding benchmark for language models. arXiv preprint arXiv:2312.04724; 2023.

[35] Shao M, *et al.* NYU CTF Bench: A Scalable Open-Source Benchmark Dataset for Evaluating LLMs in Offensive Security. arXiv preprint arXiv:2406.05590; 2024.

[36] Cheng Y, Liu Y, Li C, Song D, Gao P. CTIArena: Benchmarking LLM Knowledge and Reasoning Across Heterogeneous Cyber Threat Intelligence. arXiv preprint arXiv:2510.11974; 2025.

[37] Bhusal D, Alam MT, Nguyen L, Mahara A, Lightcap Z, Frazier R, *et al.* SECURE: Benchmarking Generative Large Language Models for Cybersecurity Advisory. arXiv preprint arXiv:2405.20441; 2024.

[38] NetSPI. Balancing Security and Usability of Large Language Models: An LLM Benchmarking Framework. 2024.

[39] Zhang AK, *et al.* Cybench: A framework for evaluating cybersecurity capabilities and risks of language models.

arXiv preprint arXiv:2408.08926; 2024.

[40] Tan RS, *et al.* LLMs in the SOC: An Empirical Study of Human-AI Collaboration in Security Operations Centres. arXiv preprint arXiv:2508.18947; 2025.

[41] Cronbach LJ, Meehl PE. Construct Validity in Psychological Tests. Psychological Bulletin. 1955;52(4):281–302. doi:10.1037/h0040957.

[42] Messick S. Validity of Psychological Assessment: Validation of Inferences from Persons' Responses and Performances as Scientific Inquiry into Score Meaning. American Psychologist. 1995;50(9):741–749. doi:10.1037/0003-066X.50.9.741.

[43] Campbell DT. Assessing the Impact of Planned Social Change. Evaluation and Program Planning. 1979;2(1):67–90. doi:10.1016/0149-7189(79)90048-X.

[44] Saltzer JH, Schroeder MD. The Protection of Information in Computer Systems. Proceedings of the IEEE. 1975;63(9):1278–1308. doi:10.1109/PROC.1975.9939.

[45] Joint Task Force Transformation Initiative. Guide for Conducting Risk Assessments. National Institute of Standards and Technology; 2012. Report No.: NIST SP 800-30, Revision 1. doi:10.6028/NIST.SP.800-30r1.

[46] Bansal G, Wu T, Zhou J, Fok R, *et al.* Does the Whole Exceed Its Parts? The Effect of AI Explanations on Complementary Team Performance. In: Proc. CHI Conference on Human Factors in Computing Systems; 2021. doi:10.1145/3411764.3445717.

[47] Buçinca Z, Malaya MB, Gajos KZ. To Trust or to Think: Cognitive Forcing Functions Can Reduce Overreliance on AI in AI-Assisted Decision-Making. Proc. ACM on Human-Computer Interaction (CSCW). 2021;5. doi:10.1145/3449287.

[48] Samvelyan M, Raparthy SC, Lupu A, Hambro E, Markosyan AH, Bhatt M, *et al.* Rainbow Teaming: Open-Ended Generation of Diverse Adversarial Prompts. In: Advances in Neural Information Processing Systems (NeurIPS); 2024.

[49] Cheng Y, Bajaber O, Tsegai SA, Song D, Gao P. CTINexus: Automatic Cyber Threat Intelligence Knowledge Graph Construction Using Large Language Models. In: Proceedings of the 2025 IEEE European Symposium on Security and Privacy (EuroS&P); 2025.

[50] Hu Y, Zou F, Han J, Sun X, Wang Y. LLM-TIKG: Threat intelligence knowledge graph construction utilizing large language model. Computers & Security. 2024;145:103999. doi:10.1016/j.cose.2024.103999.

[51] Aguilera-Martínez F, Berzal F. LLM Security: Vulnerabilities, Attacks, Defenses, and Countermeasures. arXiv preprint arXiv:2505.01177; 2025.

[52] Gong C, Li Z, Li X. Information Security Based on LLM Approaches: A Review. arXiv preprint arXiv:2507.18215; 2025.

[53] Verma A, *et al.* Recent advancements in LLM Red-Teaming: Techniques, Defenses, and Ethical Considerations. arXiv preprint arXiv:2410.09097; 2024.

[54] Deng G, *et al.* Co-RedTeam: Orchestrated Security Discovery and Exploitation with LLM Agents. arXiv preprint arXiv:2602.02164; 2026.

[55] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, *et al.* Attention is all you need. In: Advances in neural information processing systems; 2017. p. 5998–6008.

[56] Devlin J, Chang M, Lee K, Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proc. NAACL-HLT; 2019. doi:10.18653/v1/N19-1423.

[57] Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I. Language Models are Unsupervised Multitask Learners. OpenAI; 2019.

[58] Brown TB, Mann B, Ryder N, Subbiah M, *et al.* Language Models are Few-Shot Learners. In: Advances in Neural Information Processing Systems (NeurIPS); 2020.

[59] Raffel C, Shazeer N, Roberts A, Lee K, *et al.* Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Journal of Machine Learning Research. 2020;21.

[60] Lewis M, Liu Y, Goyal N, Ghazvininejad M, *et al.* BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In: Proc. ACL; 2020. doi:10.18653/v1/2020.acl-main.703.

[61] Ouyang L, Wu J, Jiang X, Almeida D, *et al.* Training Language Models to Follow Instructions with Human Feedback. In: Advances in Neural Information Processing Systems (NeurIPS); 2022.

[62] Wei J, Bosma M, Zhao VY, Guu K, *et al.* Finetuned Language Models are Zero-Shot Learners. In: Proc. ICLR; 2022.

[63] Sanh V, Webson A, Raffel C, Bach SH, *et al.* Multitask Prompted Training Enables Zero-Shot Task Generalization. In: Proc. ICLR; 2022.

[64] Chung HW, Hou L, Longpre S, Zoph B, *et al.* Scaling Instruction-Finetuned Language Models. Journal of Machine Learning Research. 2024;25.

[65] Christiano PF, Leike J, Brown T, Martic M, Legg S, Amodei D. Deep Reinforcement Learning from Human Preferences. In: Advances in Neural Information Processing Systems (NeurIPS); 2017.

[66] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347; 2017.

[67] Rafailov R, Sharma A, Mitchell E, Ermon S, Manning CD, Finn C. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In: Advances in Neural Information Processing Systems (NeurIPS); 2023.

[68] Bai Y, Kadavath S, Kundu S, *et al.* Constitutional AI: Harmlessness from AI Feedback. arXiv preprint arXiv:2212.08073; 2022.

[69] Touvron H, Lavril T, Izacard G, *et al.* Llama: Open and Efficient Foundation Language Models. arXiv preprint arXiv:2302.13971; 2023.

[70] Touvron H, Martin L, Stone K, *et al.* Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv preprint arXiv:2307.09288; 2023.

[71] Chowdhery A, Narang S, Devlin J, *et al.* PaLM: Scaling Language Modeling with Pathways. Journal of Machine Learning Research. 2023;24.

[72] Hu EJ, Shen Y, Wallis P, Allen-Zhu Z, *et al.* LoRA: Low-Rank Adaptation of Large Language Models. In: Proc. ICLR; 2022.

[73] Dettmers T, Pagnoni A, Holtzman A, Zettlemoyer L. QLoRA: Efficient Finetuning of Quantized LLMs. In: Advances in Neural Information Processing Systems (NeurIPS); 2023.

[74] Kaplan J, McCandlish S, Henighan T, Brown TB, *et al.* Scaling Laws for Neural Language Models. arXiv preprint arXiv:2001.08361; 2020.

[75] Hoffmann J, Borgeaud S, Mensch A, *et al.* Training Compute-Optimal Large Language Models. In: Advances in Neural Information Processing Systems (NeurIPS); 2022.

[76] Wei J, Tay Y, Bommasani R, Raffel C, *et al.* Emergent Abilities of Large Language Models. Transactions on Machine Learning Research. 2022.

[77] Bubeck S, Chandrasekaran V, Eldan R, *et al.* Sparks of Artificial General Intelligence: Early Experiments with GPT-4. arXiv preprint arXiv:2303.12712; 2023.

[78] Bommasani R, Hudson DA, Adeli E, Altman R, *et al.* On the Opportunities and Risks of Foundation Models. arXiv preprint arXiv:2108.07258; 2021.

[79] OpenAI. GPT-4 Technical Report. arXiv preprint arXiv:2303.08774; 2023.

[80] Wang A, Pruksachatkun Y, Nangia N, Singh A, *et al.* SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In: Advances in Neural Information Processing Systems (NeurIPS); 2019.

[81] Srivastava A, Rastogi A, Rao A, *et al.* Beyond the Imitation Game: Quantifying and Extrapolating the Capabilities of Language Models. Transactions on Machine Learning Research. 2023.

[82] Zhong W, Cui R, Guo Y, Liang Y, *et al.* AGIEval: A Human-Centric Benchmark for Evaluating Foundation Models. arXiv preprint arXiv:2304.06364; 2023.

[83] Wang Y, Ma X, Zhang G, Ni Y, *et al.* MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark. Advances in Neural Information Processing Systems (NeurIPS). 2024.

[84] Wei J, Wang X, Schuurmans D, Bosma M, *et al.* Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In: Advances in Neural Information Processing Systems (NeurIPS); 2022.

[85] Cobbe K, Kosaraju V, Bavarian M, *et al.* Training Verifiers to Solve Math Word Problems. arXiv preprint arXiv:2110.14168; 2021.

[86] Hendrycks D, Burns C, Kadavath S, Arora A, *et al.* Measuring Mathematical Problem Solving with the MATH Dataset. In: Proceedings of the NeurIPS Datasets and Benchmarks Track; 2021.

[87] Chen M, Tworek J, Jun H, Yuan Q, *et al.* Evaluating Large Language Models Trained on Code. arXiv preprint arXiv:2107.03374; 2021.

[88] Austin J, Odena A, Nye M, Bosma M, *et al.* Program Synthesis with Large Language Models. arXiv preprint arXiv:2108.07732; 2021.

[89] Lin S, Hilton J, Evans O. TruthfulQA: Measuring How Models Mimic Human Falsehoods. In: Proc. ACL; 2022. doi:10.18653/v1/2022.acl-long.229.

[90] Zellers R, Holtzman A, Bisk Y, Farhadi A, Choi Y. HellaSwag: Can a Machine Really Finish Your Sentence? In: Proc. ACL; 2019. doi:10.18653/v1/P19-1472.

[91] Guha N, Nyarko J, Ho DE, Ré C, *et al.* LegalBench: A Collaboratively Built Benchmark for Measuring Legal Reasoning in Large Language Models. Advances in Neural Information Processing Systems (NeurIPS). 2023.

[92] Jin Q, Dhingra B, Liu Z, Cohen WW, Lu X. PubMedQA: A Dataset for Biomedical Research Question Answering. Proc. EMNLP-IJCNLP. 2019. doi:10.18653/v1/D19-1259.

[93] Zheng L, Chiang W, Sheng Y, Zhuang S, *et al.* Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In: Advances in Neural Information Processing Systems (NeurIPS); 2023.

[94] Min S, Krishna K, Lyu X, Lewis M, *et al.* FActScore: Fine-Grained Atomic Evaluation of Factual Precision in Long Form Text Generation. In: Proc. EMNLP; 2023.

[95] Manakul P, Liusie A, Gales MJ. SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models. In: Proc. EMNLP; 2023.

[96] Lewis P, Perez E, Piktus A, Petroni F, *et al.* Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In: Advances in Neural Information Processing Systems (NeurIPS); 2020.

[97] Guu K, Lee K, Tung Z, Pasupat P, Chang M. REALM: Retrieval-Augmented Language Model Pre-Training. In: Proc. ICML; 2020.

[98] Borgeaud S, Mensch A, Hoffmann J, Cai T, *et al.* Improving Language Models by Retrieving from Trillions of Tokens. In: Proc. ICML; 2022.

[99] Izacard G, Lewis P, Lomeli M, Hosseini L, *et al.* Atlas: Few-shot Learning with Retrieval Augmented Language Models. Journal of Machine Learning Research. 2023;24.

[100] Karpukhin V, Oguz B, Min S, Lewis P, *et al.* Dense Passage Retrieval for Open-Domain Question Answering. In: Proc. EMNLP; 2020. doi:10.18653/v1/2020.emnlp-main.550.

[101] Khattab O, Zaharia M. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In: Proc. ACM SIGIR; 2020. doi:10.1145/3397271.3401075.

[102] Johnson J, Douze M, Jégou H. Billion-Scale Similarity Search with GPUs. IEEE Transactions on Big Data. 2021;7(3). doi:10.1109/TBDATA.2019.2921572.

[103] Malkov YA, Yashunin DA. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2020;42(4). doi:10.1109/TPAMI.2018.2889473.

[104] Asai A, Wu Z, Wang Y, Sil A, Hajishirzi H. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In: Proc. ICLR; 2024.

[105] Shi W, Min S, Yasunaga M, Seo M, *et al.* REPLUG: Retrieval-Augmented Black-Box Language Models. In: Proc. NAACL; 2024.

[106] Gao Y, Xiong Y, Gao X, Jia K, *et al.* Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv preprint arXiv:2312.10997; 2023.

[107] Zhong Z, Huang Z, Wettig A, Chen D. Poisoning Retrieval Corpora by Injecting Adversarial Passages. Proc. EMNLP. 2023.

[108] Chaudhari H, Severi G, Abascal J, Jagielski M, *et al.* Phantom: General Trigger Attacks on Retrieval Augmented Language Generation. arXiv preprint arXiv:2405.20485; 2024.

[109] Xiang C, Wu T, Zhong Z, Wagner D, Chen D, Mittal P. Certifiably Robust RAG Against Retrieval Corruption. arXiv preprint arXiv:2405.15556; 2024.

[110] Yao S, Zhao J, Yu D, Du N, *et al.* ReAct: Synergizing Reasoning and Acting in Language Models. In: Proc. ICLR; 2023.

[111] Schick T, Dwivedi-Yu J, Dessì R, Raileanu R, *et al.* Toolformer: Language Models Can Teach Themselves to Use Tools. In: Advances in Neural Information Processing Systems (NeurIPS); 2023.

[112] Shinn N, Cassano F, Gopinath A, Narasimhan K, Yao S. Reflexion: Language Agents with Verbal Reinforcement Learning. In: Advances in Neural Information Processing Systems (NeurIPS); 2023.

[113] Park JS, O'Brien J, Cai CJ, Morris MR, *et al.* Generative Agents: Interactive Simulacra of Human Behavior. In: Proc. ACM UIST; 2023. doi:10.1145/3586183.3606763.

[114] Qin Y, Liang S, Ye Y, Zhu K, *et al.* ToolLLM: Facilitating Large Language Models to Master 16000+ Real-World APIs. In: Proc. ICLR; 2024.

[115] Wu Q, Bansal G, Zhang J, Wu Y, *et al.* AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. arXiv preprint arXiv:2308.08155; 2023.

[116] Hong S, Zhuge M, Chen J, Zheng X, *et al.* MetaGPT: Meta Programming for a Multi-Agent Collaborative Framework. In: Proc. ICLR; 2024.

[117] Xi Z, Chen W, Guo X, He W, *et al.* The Rise and Potential of Large Language Model Based Agents: A Survey. arXiv preprint arXiv:2309.07864; 2023.

[118] Liu X, Yu H, Zhang H, Xu Y, *et al.* AgentBench: Evaluating LLMs as Agents. In: Proc. ICLR; 2024.

[119] Fang R, Bindu R, Gupta A, Zhan Q, Kang D. LLM Agents can Autonomously Hack Websites. arXiv preprint arXiv:2402.06664; 2024.

[120] Xu J, Stokes JW, McDonald G, Bai X, *et al.* AutoAttacker: A Large Language Model Guided System to Implement Automatic Cyber-attacks. arXiv preprint arXiv:2403.01038; 2024.

[121] Happe A, Cito J. Getting Pwn'd by AI: Penetration Testing with Large Language Models. Proc. 31st ACM Joint European Software Engineering Conference and Symposium (ESEC/FSE). 2023. doi:10.1145/3611643.3613083.

[122] Zhan Q, Liang Z, Ying Z, Kang D. InjecAgent: Benchmarking

Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. Findings of ACL. 2024.

[123] Debenedetti E, Zhang J, Balunovic M, Beurer-Kellner L, *et al*. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. In: Proceedings of the NeurIPS Datasets and Benchmarks Track; 2024.

[124] Deng G, *et al*. A Survey on Agentic Security: Applications, Threats and Defenses. arXiv preprint arXiv:2510.06445; 2025.

[125] Wei A, Haghtalab N, Steinhardt J. Jailbroken: How Does LLM Safety Training Fail? In: Advances in Neural Information Processing Systems (NeurIPS); 2023.

[126] Chao P, Robey A, Dobriban E, Hassani H, Pappas GJ, Wong E. Jailbreaking Black Box Large Language Models in Twenty Queries. In: Proceedings of the IEEE Conference on Secure and Trustworthy Machine Learning (SaTML); 2025.

[127] Mehrotra A, Zampetakis M, Kassianik P, Nelson B, *et al.* Tree of Attacks: Jailbreaking Black-Box LLMs Automatically. In: Advances in Neural Information Processing Systems (NeurIPS); 2024.

[128] Liu X, Xu N, Chen M, Xiao C. AutoDAN: Generating Stealthy Jailbreak Prompts on Aligned Large Language Models. In: Proc. ICLR; 2024.

[129] Shen X, Chen Z, Backes M, Shen Y, Zhang Y. "Do Anything Now": Characterizing and Evaluating In-the-Wild Jailbreak Prompts on Large Language Models. In: Proc. ACM CCS; 2024.

[130] Andriushchenko M, Croce F, Flammarion N. Jailbreaking Leading Safety-Aligned LLMs with Simple Adaptive Attacks. Proc. ICLR. 2025.

[131] Zeng Y, Lin H, Zhang J, Yang D, Jia R, Shi W. How Johnny Can Persuade LLMs to Jailbreak Them: Rethinking Persuasion to Challenge AI Safety by Humanizing LLMs. Proc. ACL. 2024.

[132] Ren Q, Li H, Liu D, Xie Z, *et al.* Derail Yourself: Multi-turn LLM Jailbreak Attack through Self-discovered Clues. arXiv preprint arXiv:2410.10700; 2024.

[133] Schulhoff S, Pinto J, Khan A, Bouchard L, *et al.* Ignore This Title and HackAPrompt: Exposing Systemic Vulnerabilities of LLMs Through a Global Prompt Hacking Competition. Proc. EMNLP. 2023.

[134] Greshake K, Abdelnabi S, Mishra S, Endres C, Holz T, Fritz M. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In: Proc. 16th ACM Workshop on Artificial Intelligence and Security (AISec); 2023.

[135] Abdelnabi S, Greshake K, Mishra S, *et al.* More than You've Asked For: A Comprehensive Analysis of Novel Prompt Injection Threats to Application-Integrated Large Language Models. arXiv preprint arXiv:2302.12173; 2023.

[136] Liu Y, Jia Y, Geng R, Jia J, Gong NZ. Formalizing and Benchmarking Prompt Injection Attacks and Defenses. In: Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24); 2024.

[137] Yi J, Xie Y, Zhu B, Hines K, *et al*. Benchmarking and Defending Against Indirect Prompt Injection Attacks on Large Language Models. arXiv preprint arXiv:2312.14197; 2024.

[138] Carlini N, Tramer F, Wallace E, Jagielski M, *et al.* Extracting Training Data from Large Language Models. In: Proceedings of the 30th USENIX Security Symposium (USENIX Security 21); 2021.

[139] Carlini N, Ippolito D, Jagielski M, Lee K, Tramer F, Zhang C. Quantifying Memorization Across Neural Language Models. In: Proc. ICLR; 2023.

[140] Wallace E, Feng S, Kandpal N, Gardner M, Singh S. Universal Adversarial Triggers for Attacking and Analyzing NLP. In: Proc. EMNLP-IJCNLP; 2019. doi:10.18653/v1/D19-1221.

[141] Ganguli D, Lovitt L, Kernion J, Askell A, *et al.* Red Teaming Language Models to Reduce Harms: Methods, Scaling Behaviors, and Lessons Learned. arXiv preprint arXiv:2209.07858; 2022.

[142] Perez E, Huang S, Song F, Cai T, *et al.* Red Teaming Language Models with Language Models. In: Proc. EMNLP; 2022. doi: 10.18653/v1/2022.emnlp-main.225.

[143] Yu J, Lin X, Yu Z, Xing X. GPTFUZZER: Red Teaming Large Language Models with Auto-Generated Jailbreak Prompts. arXiv preprint arXiv:2309.10253; 2023.

[144] Lin L, Mu H, Zhai Z, Wang M, *et al.* Large Language Model Red Teaming: A Survey of Attacks, Defenses, and Evaluation. arXiv preprint arXiv:2407.14937; 2024.

[145] Mazeika M, Phan L, Yin X, Zou A, *et al.* HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal. Proc. ICML. 2024.

[146] Xu Z, Liu Y, Deng G, Li Y, Picek S. A Comprehensive Study of Jailbreak Attack versus Defense for Large Language Models. Findings of ACL. 2024.

[147] Jain N, Schwarzschild A, Wen Y, Somepalli G, *et al.* Baseline Defenses for Adversarial Attacks Against Aligned Language Models. arXiv preprint arXiv:2309.00614; 2023.

[148] Robey A, Wong E, Hassani H, Pappas GJ. SmoothLLM: Defending Large Language Models Against Jailbreaking Attacks. arXiv preprint arXiv:2310.03684; 2023.

[149] Kumar A, Agarwal C, Srinivas S, Feizi S, Lakkaraju H. Certifying LLM Safety Against Adversarial Prompting. arXiv preprint arXiv:2309.02705; 2023.

[150] Inan H, Upasani K, Chi J, Rungta R, *et al.* Llama Guard: LLM-based Input-Output Safeguard for Human-AI Conversations. arXiv preprint arXiv:2312.06674; 2023.

[151] Weidinger L, Mellor J, Rauh M, Griffin C, *et al.* Ethical and Social Risks of Harm from Language Models. arXiv preprint arXiv:2112.04359; 2021.

[152] Ji Z, Lee N, Frieske R, Yu T, *et al.* Survey of Hallucination in Natural Language Generation. ACM Computing Surveys. 2023;55(12). doi:10.1145/3571730.

[153] Huang L, Yu W, Ma W, Zhong W, *et al.* A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. ACM Transactions on Information Systems. 2025;43(2).

[154] Sommer R, Paxson V. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In: Proceedings of the IEEE Symposium on Security and Privacy; 2010. doi:10.1109/SP.2010.25.

[155] Arp D, Quiring E, Pendlebury F, Warnecke A, *et al.* Dos and Don'ts of Machine Learning in Computer Security. In: Proceedings of the 31st USENIX Security Symposium (USENIX Security 22); 2022.

[156] Apruzzese G, Laskov P, Oca EM, Mallouli W, *et al.* The Role of Machine Learning in Cybersecurity. In: Proceedings of the Digital Threats: Research and Practice; 2023.

[157] Tavallaee M, Bagheri E, Lu W, Ghorbani AA. A Detailed Analysis of the KDD CUP 99 Data Set. In: Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA); 2009. doi:10.1109/CISDA.2009.5356528.

[158] Moustafa N, Slay J. UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems. In: Proceedings of the Military Communications and Information Systems Conference (MilCIS); 2015. doi:10.1109/MilCIS.2015.7348942.

[159] Sharafaldin I, Lashkari AH, Ghorbani AA. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In: Proc. ICISSP; 2018. doi:10.5220/0006639801080116.

[160] Anderson HS, Roth P. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. arXiv preprint arXiv:1804.04637; 2018.

[161] Raff E, Barker J, Sylvester J, Brandon R, Catanzaro B,

Nicholas CK. Malware Detection by Eating a Whole EXE. In: Proceedings of the AAAI Workshop on Artificial Intelligence for Cyber Security; 2018.

[162] Downing E, Mirsky Y, Park K, Lee W. DeepReflect: Discovering Malicious Functionality through Binary Reconstruction. In: Proceedings of the 30th USENIX Security Symposium (USENIX Security 21); 2021.

[163] Buczak AL, Guven E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. IEEE Communications Surveys & Tutorials. 2016;18(2). doi:10.1109/COMST.2015.2494502.

[164] Ucci D, Aniello L, Baldoni R. Survey of Machine Learning Techniques for Malware Analysis. Computers & Security. 2019;81. doi: 10.1016/j.cose.2018.11.001.

[165] Quiring E, Klein D, Arp D, Johns M, Rieck K. Adversarial Preprocessing: Understanding and Preventing Image-Scaling Attacks in Machine Learning. In: Proceedings of the 29th USENIX Security Symposium (USENIX Security 20); 2020.

[166] Sahingoz OK, Buber E, Demir O, Diri B. Machine Learning Based Phishing Detection from URLs. Expert Systems with Applications. 2019;117. doi: 10.1016/j.eswa.2018.09.029.

[167] Li Z, Zou D, Xu S, Ou X, Jin H, *et al.* VulDeePecker: A Deep Learning-Based System for Vulnerability Detection. In: Proc. Network and Distributed System Security Symposium (NDSS); 2018.

[168] Zhou Y, Liu S, Siow J, Du X, Liu Y. Devign: Effective Vulnerability Identification by Learning Comprehensive Program Semantics via Graph Neural Networks. In: Advances in Neural Information Processing Systems (NeurIPS); 2019.

[169] Chakraborty S, Krishna R, Ding Y, Ray B. Deep Learning Based Vulnerability Detection: Are We There Yet? IEEE Transactions on Software Engineering. 2022;48(9). doi:10.1109/TSE.2021.3078750.

[170] Fu M, Tantithamthavorn C. LineVul: A Transformer-Based Line-Level Vulnerability Prediction. In: Proc. 19th International Conference on Mining Software Repositories (MSR); 2022. doi:10.1145/3524842.3528452.

[171] Fan J, Li Y, Wang S, Wang G, Chen X. BigVul: A large-scale, fine-grained vulnerability dataset. arXiv preprint arXiv:2010.08772; 2020.

[172] Feng Z, Guo D, Tang D, Duan N, *et al.* CodeBERT: A Pre-Trained Model for Programming and Natural Languages. In: Findings of EMNLP; 2020. doi: 10.18653/v1/2020.findings-emnlp.139.

[173] Guo D, Ren S, Lu S, Feng Z, *et al.* GraphCodeBERT: Pre-training Code Representations with Data Flow. In: Proc. ICLR; 2021.

[174] Wang Y, Wang W, Joty S, Hoi SC. CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation. In: Proc. EMNLP; 2021. doi: 10.18653/v1/2021.emnlp-main.685.

[175] Nijkamp E, Pang B, Hayashi H, Tu L, *et al.* CodeGen: An Open Large Language Model for Code with Multi-Turn Program Synthesis. In: Proc. ICLR; 2023.

[176] Li R, Allal LB, Zi Y, Muennighoff N, *et al.* StarCoder: May the Source Be with You! Transactions on Machine Learning Research. 2023.

[177] Rozière B, Gehring J, Gloeckle F, Sootla S, *et al.* Code Llama: Open Foundation Models for Code. arXiv preprint arXiv:2308.12950; 2023.

[178] Pearce H, Ahmad B, Tan B, Dolan-Gavitt B, Karri R. Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions. In: Proceedings of the IEEE Symposium on Security and Privacy; 2022.

[179] Pearce H, Tan B, Ahmad B, Karri R, Dolan-Gavitt B. Examining Zero-Shot Vulnerability Repair with Large Language Models. In: Proceedings of the IEEE Symposium on Security and Privacy; 2023.

[180] Sandoval G, Pearce H, Nys T, Karri R, Garg S, Dolan-Gavitt B. Lost at C: A User Study on the Security Implications of Large Language Model Code Assistants. In: Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23); 2023.

[181] Tihanyi N, Bisztray T, Jain R, Ferrag MA, *et al.* The FormAI Dataset: Generative AI in Software Security through the Lens of Formal Verification. Proc. 19th International Conference on Predictive Models and Data Analytics in Software Engineering. 2023. doi:10.1145/3617555.3617874.

[182] Khare A, Dutta S, Li Z, Solko-Breslin A, *et al.* Can ChatGPT Pass the Vulnerability Detection Test? An Empirical Study. arXiv preprint arXiv:2311.16169; 2023.

[183] Husari G, Al-Shaer E, Ahmed M, Chu B, Niu X. TTPDrill: Automatic and Accurate Extraction of Threat Actions from Unstructured Text of CTI Sources. In: Proc. 33rd Annual Computer Security Applications Conference (ACSAC); 2017.

[184] Liao X, Yuan K, Wang X, Li Z, Xing L, Beyah R. Acing the IOC Game: Toward Automatic Discovery and Analysis of Open-Source Cyber Threat Intelligence. In: Proc. ACM CCS; 2016. doi:10.1145/2976749.2978315.

[185] Li Z, Zeng J, Chen Y, Liang Z. AttacKG: Constructing Technique Knowledge Graph from Cyber Threat Intelligence Reports. In: Proc. European Symposium on Research in Computer Security (ESORICS); 2022.

[186] Gao P, Shao F, Liu X, Xiao X, *et al.* Enabling Efficient Cyber Threat Hunting with Cyber Threat Intelligence. In: Proc. IEEE 37th International Conference on Data Engineering (ICDE); 2021.

[187] Milajerdi SM, Gjomemo R, Eshete B, Sekar R, Venkatakrishnan V. HOLMES: Real-Time APT Detection through Correlation of Suspicious Information Flows. In: Proceedings of the IEEE Symposium on Security and Privacy; 2019.

[188] Milajerdi SM, Eshete B, Gjomemo R, Venkatakrishnan V. POIROT: Aligning Attack Behavior with Kernel Audit Records for Cyber Threat Hunting. In: Proc. ACM CCS; 2019.

[189] Strom BE, Applebaum A, Miller DP, Nickels KC, *et al.* MITRE ATT&CK: Design and Philosophy. The MITRE Corporation; 2018.

[190] Lai V, Chen C, Liao QV, Smith-Renner A, Tan C. Towards a Science of Human-AI Decision Making: A Survey of Empirical Studies. arXiv preprint arXiv:2112.11471; 2021.

[191] Alahmadi BA, Axon L, Martinovic I. 99% False Positives: A Qualitative Study of SOC Analysts' Perspectives on Security Alarms. In: Proceedings of the 31st USENIX Security Symposium (USENIX Security 22); 2022.

[192] Hassan WU, Guo S, Li D, Chen Z, *et al.* NoDoze: Combatting Threat Alert Fatigue with Automated Provenance Triage. In: Proc. Network and Distributed System Security Symposium (NDSS); 2019.

[193] Vielberth M, Bohm F, Fichtinger I, Pernul G. Security Operations Center: A Systematic Study and Open Challenges. IEEE Access. 2020;8. doi:10.1109/ACCESS.2020.3045514.

[194] Yao Y, Duan J, Xu K, Cai Y, Sun Z, Zhang Y. A Survey on Large Language Model (LLM) Security and Privacy: The Good, the Bad, and the Ugly. High-Confidence Computing. 2024;4(2).

[195] Motlagh FN, Hajizadeh M, Majd M, Najafi P, *et al.* Large Language Models in Cybersecurity: State-of-the-Art. arXiv preprint arXiv:2402.00891; 2024.

[196] Xu H, Wang S, Li N, Wang K, *et al.* Large Language Models for Cyber Security: A Systematic Literature Review. arXiv preprint arXiv:2405.04760; 2024.

[197] Hassanin M, Moustafa N. A Comprehensive Overview of Large Language Models (LLMs) for Cyber Defences: Opportunities and Directions. arXiv preprint arXiv:2405.14487; 2024.

[198] Ferrag MA, Alwahedi F, Battah A, Cherif B, *et al.* Generative

AI and Large Language Models for Cyber Security: All Insights You Need. arXiv preprint arXiv:2405.12750; 2024.

[199] Zhang J, Bu H, Wen H, Liu Y, *et al.* When LLMs Meet Cybersecurity: A Systematic Literature Review. Cybersecurity. 2025;8.

[200] Das BC, Amini MH, Wu Y. A Survey on Large Language Model (LLM) Security: Threats, Vulnerabilities, and Defenses. ACM Computing Surveys. 2025.

[201] Happe A, Cito J. Benchmarking Practices in LLM-driven Offensive Security: Testbeds, Metrics, and Experiment Design. arXiv preprint arXiv:2504.10112; 2025.

Received on 15-07-2026 Accepted on 17-08-2026 Published on 07-09-2026

https://doi.org/10.65879/3070-5789.2026.02.05