International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(16s) 2026 pp. 1704-1718
https://doi.org/10.70917/ijcisim-2026-4703
Article

Defensive Reverse Engineering of LLM Applications: A Black-Box Framework for Security Risk Scoring and Mitigation

Bhavesh B. Prajapati[1*], Bhavya Shah[2†]

[*1]IT Department, L. D. College of Engineering, Commissionerate of Technical Education, Government of Gujarat, India
Email: b.b.prajapati@gmail.com, ORCID: 0000-0002-8015-7934
[†2]Individual Scholar, Email: shahbhavya5800@gmail.com

Abstract: Large language model (LLM) applications now combine hidden prompts, retrieval pipelines, memory stores, content filters, tool calls, delegated identities, and downstream automation. Security reviewers are increasingly asked to assess such systems without access to source code, model weights, prompt templates, vector-store configuration, or internal logs. This paper presents D-RELLM, a defensive reverse-engineering framework for black-box security assessment of deployed LLM applications. The framework maps observable application behavior to an attack-surface graph, executes bounded and non-destructive probes, converts evidence into reproducible risk scores, and links each finding to mitigation and re-test criteria. Unlike model-only jailbreak evaluation, D-RELLM treats the deployed application as a socio-technical system whose risk depends on instruction hierarchy, retrieval trust, authorization, tool agency, output handling, monitoring, and operational controls. The paper defines a threat model, a probe taxonomy, a weighted scoring equation, a confidence equation, an application-level aggregation method, an evidence schema, and a mitigation playbook. A synthetic pilot across three representative archetypes—chatbot, retrieval-augmented generation (RAG) assistant, and tool-using agent—illustrates how direct prompt injection, indirect prompt injection, retrieval poisoning exposure, excessive agency, output-handling weaknesses, and sensitive-information disclosure can be prioritized before and after remediation. The result is a practical, auditable, and publication-ready method for defenders who need to quantify residual LLM application risk from black-box evidence while avoiding unsafe exploitation.
Keywords: large language models, black-box testing, prompt injection, AI agents, retrieval-augmented generation, risk scoring, red teaming, secure AI, defensive reverse engineering

1. INTRODUCTION

LLM applications are no longer isolated chat boxes. A production assistant can receive a system prompt from an orchestrator, retrieve enterprise records, call functions with a user's identity, summarize tool output, store memory, render HTML or code, and pass structured results into a downstream workflow. This integration makes the application boundary more important than the model boundary: the same base model can be low risk in a static question-answering interface and high risk when connected to email, tickets, file systems, payments, or administrative APIs.

The security challenge is that defenders often see only the outside of the system. A third-party assessment, procurement review, bug-bounty triage, compliance audit, or internal red-team exercise may provide only a web interface or API key.

The reviewer may not know the prompt hierarchy, the retriever's access-control policy, the model version, the content-filter chain, or the exact tool permissions. This situation resembles reverse engineering, but the objective is

defensive: infer security-relevant design properties from observable behavior and then recommend controls without extracting secrets, damaging systems, or disclosing private data.

Existing guidance provides strong foundations. OWASP identifies LLM-specific application risks, NIST frames generative-AI risk management, MITRE ATLAS maps adversary tactics against AI systems, and CVSS/FAIR-style approaches help communicate severity [1]–[13]. Research has demonstrated direct prompt injection, indirect prompt injection, jailbreak automation, retrieval poisoning, model extraction, training-data leakage, and guardrail limitations [14]–[26]. However, defenders still lack a compact black-box method that (i) models the entire deployed application, (ii) separates safe probes from harmful exploit activity, (iii) assigns comparable risk scores, and (iv) validates mitigation through repeatable evidence.

This paper makes four contributions. First, it defines D-RELLM, a black-box defensive reverse-engineering workflow for LLM applications. Second, it introduces a quantitative scoring model that combines exploitability, impact, exposure, agency, control bypass, detectability, and mitigation strength. Third, it proposes an evidence record that supports reproducibility without publishing sensitive prompts or data. Fourth, it maps common black-box findings to engineering controls and residual-risk tests. The method is intentionally conservative: it helps authorized reviewers reason about security risk without providing operational exploit recipes.

2. RELATED WORK AND SECURITY CONTEXT

A. LLM Application Security

Prompt injection research established that natural-language inputs can conflict with developer intent. Direct prompt injection targets the user's active instruction channel, while indirect prompt injection hides adversarial instructions in web pages, documents, emails, or retrieved records that the LLM later reads [14], [15]. Formal frameworks and defenses such as structured queries, instruction hierarchy training, and agent-specific benchmarks show that robustness depends on how an application separates trusted instructions from untrusted content [16]–[19], [27], [28]. Red-team tools and benchmarks such as garak, PyRIT, CyberSecEval, HackAPrompt, Tensor Trust, PAIR, TAP, GPTFuzz, AutoDAN, MasterKey, and universal adversarial suffixes demonstrate why black-box exploration must be expected in practice [20]–[23], [29]–[40].

B. RAG, Agents, and Tool Use

RAG systems and agents expand the security boundary. Dense retrieval, vector indexes, rerankers, citation generation, and tool calls create paths through which untrusted content can influence privileged behavior [41]–[47]. Recent work on poisoned retrieval, confused-deputy behavior, and backdoored retrievers highlights that a correct base model can still be unsafe when the application treats data as instructions [48]–[51]. Tool-use systems such as ReAct, MRKL, Toolformer, Gorilla, WebGPT, and API-calling agents motivate testing confirmation gates, least-privilege tokens, dry-run modes, and tool-result quarantine [52]–[58].

C. Model Evaluation, Privacy, and Assurance

General LLM research explains why behavior varies by model, instruction tuning, reinforcement learning from human feedback, and benchmark domain [59]–[70]. Capability and safety benchmarks provide useful but incomplete context for deployed application risk [71]–[83]. Privacy and black-box ML security literature motivates treating leakage, repeated-query behavior, and extraction-like signals as evidence, while adversarial-learning work warns that narrow defenses can create a false sense of security [84]–[103]. Guardrail systems, model cards, datasheets, audits, and AI-risk taxonomies further inform the mitigation layer of D-RELLM [104]–[120].

3. THREAT MODEL, ETHICS, AND ASSUMPTIONS

A. Access Levels

D-RELLM assumes an authorized reviewer with one of three black-box access levels. Public access allows interaction as an unauthenticated or trial user. Tenant access allows testing within a controlled organization account with synthetic data and test identities. Sandbox-admin access allows the reviewer to create benign canary documents, test tool endpoints, and synthetic users, but still not inspect internal prompts, weights, source code, or logs unless the system exposes them. Findings must be interpreted relative to the access level because a failure reachable only by an administrator has different exploitability than one reachable by any user.

B. Security Objectives and Non-goals

The framework measures confidentiality, integrity, availability, authorization, safety, and financial exposure. It specifically looks for leakage of prompts or private data, corruption of generated answers, unauthorized tool action, cross-tenant retrieval, unsafe downstream output, excessive cost consumption, and lack of detection. It does not attempt credential theft, production data exfiltration, destructive tool invocation, bypass of payment controls, or unauthorized scanning. High-impact actions are replaced with canaries, dry-run calls, or sandbox resources. This makes the method suitable for internal security review, procurement validation, and responsible disclosure.

C. Evidence Standard

A black-box LLM finding is not just a surprising transcript. D-RELLM requires four evidence elements: the observable precondition, the non-sensitive probe class, the security-relevant outcome, and a reproducibility estimate. Raw prompts that contain secrets or bypass details should not be published. Instead, reports use transcript hashes, timestamps, model/application version strings when visible, synthetic canary identifiers, and redacted screenshots. Table I defines the minimum evidence record.

TABLE I

MINIMUM EVIDENCE RECORD FOR A BLACK-BOX LLM FINDING

Field	Purpose
Scope tag	Identifies authorized tenant, test account, or sandbox resource.
Surface	UI, API, upload, URL fetcher, RAG corpus, memory, or tool interface.
Probe class	Risk category without revealing unsafe payload details.
Canary	Harmless marker used to prove influence, leakage, or tool action.
Outcome	Observable behavior, returned data class, or attempted tool call.
Replays	Number of successful repetitions over total bounded attempts.
Controls seen	Refusal, confirmation, citation, rate limit, schema validation, logging.
Hash	Digest of full transcript stored internally for audit integrity.

4. THE D-RELLM FRAMEWORK

Fig. 1 summarizes the assessment loop. The process starts with authorization and scope constraints, builds a behavioral map, generates safe probes, clusters observations, scores findings, maps controls, and replays mitigated variants. The loop is iterative because new observations often reveal hidden components such as a retriever, memory layer, or function caller.

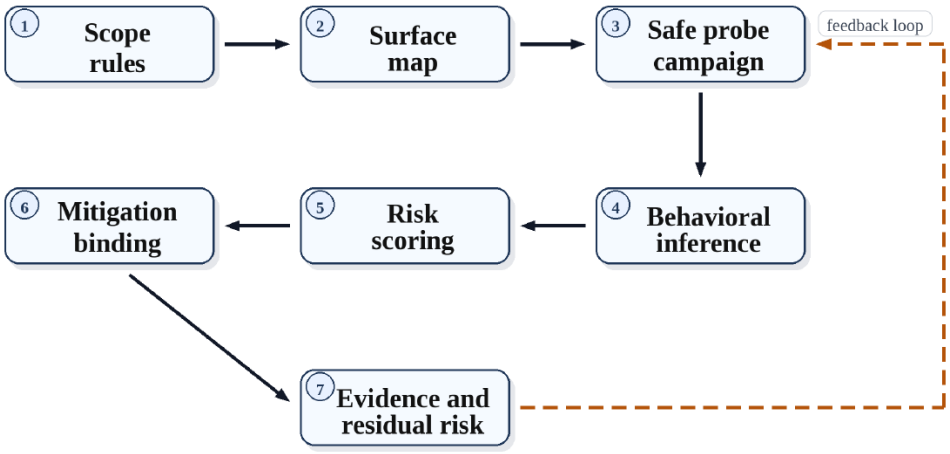

Fig. 1. D-RELLM black-box assessment loop redesigned with routed connectors and a visible residual-risk feedback path. Each iteration is bounded by authorization, synthetic data, rate limits, and non-destructive canaries.

A. Surface Mapping

Surface mapping documents visible inputs, outputs, trust boundaries, and implicit privileges. The reviewer records whether the system accepts free text, files, images, URLs, browser context, voice, email, tickets, spreadsheet rows, or API calls. The reviewer also observes citations, refusal language, memory behavior, tool traces, latency jumps, file names, generated links, and output formats. These signals are used to infer components without demanding internal documentation.

Fig. 2 expresses the inferred application as a trust-boundary graph. Edges are not attacks; they are possible influence paths. A path from untrusted content to privileged action is high priority even if no harmful action is attempted during testing. The graph helps reviewers explain why a RAG poisoning signal is more serious when the same assistant also has write-capable tools.

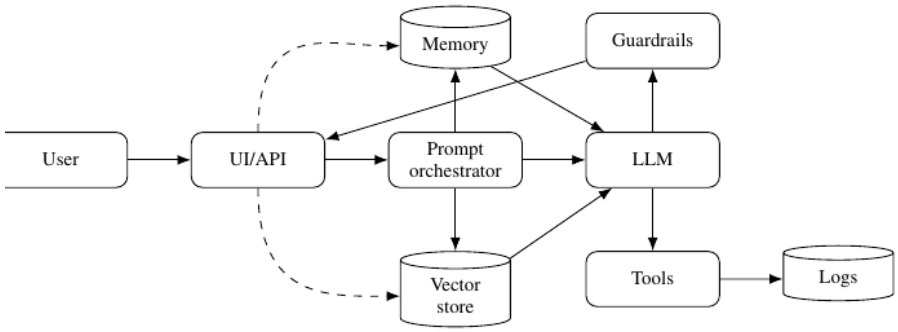

Fig. 2. Trust-boundary graph used during black-box reverse engineering. Dashed edges represent inferred content-ingestion paths such as uploads, URLs, and memory writes.

B. Probe Taxonomy

Table II lists the core probe families. The table intentionally describes defensive objectives and observable signals rather than exploit payloads. Every probe should have a benign purpose, a canary, an expected safe behavior, and a stop condition. For example, an indirect-injection probe can place a harmless marker in a sandbox document and verify whether that marker inappropriately changes the assistant's task behavior. A tool-agency probe can use a dry-run endpoint and verify whether the assistant requires confirmation before a write-like operation.

TABLE II

BLACK-BOX PROBE FAMILIES AND DEFENSIVE INTERPRETATION

Probe family	Observable evidence	Risk indicated	Mitigation direction
Direct prompt injection	Follows conflicting user instruction over stated task or role	Weak instruction priority; policy bypass	Instruction hierarchy, structured prompt channels, refusal tuning, input classification
Indirect prompt injection	Tool/RAG content changes behavior beyond retrieval task	Untrusted data treated as executable instruction	Data/instruction separation, retrieval sanitization, content provenance, tool-output quarantine
System prompt leakage	Reveals hidden policies, tool names, internal routing, or secret-like content	Confidentiality failure and reconnaissance acceleration	Secret removal from prompts, prompt minimization, canary detection, response filtering
Sensitive data disclosure	Returns another user/tenant's data, private snippets, or memory from the wrong context	Broken authorization, memory isolation, or retrieval ACLs	Per-document ACL enforcement, tenant isolation, least-privilege retrieval, audit logging
Excessive agency	Takes, schedules, or strongly plans unsafe tool action without confirmation	Confused deputy and high-impact automation	Human approval, scoped tokens, transaction limits, reversible actions, dry-run defaults
Improper output handling	Emits script, SQL, HTML, formula, or shell-like output consumed downstream	Cross-layer injection into web, database, spreadsheet, or automation systems	Output encoding, schema validation, sandboxing, downstream allowlists
Availability and cost	Long loops, repeated tool calls, oversized contexts, or retry storms	Unbounded consumption and denial of wallet/service	Quotas, circuit breakers, max tool-call depth, caching, anomaly alerts
RAG integrity	Citation mismatch, poisoned citation dominance, stale source use, or unsupported claims	Knowledge corruption and misinformation	Source ranking hardening, signed corpora, freshness checks, multi-source corroboration
Monitoring gap	No visible alert, trace identifier, or block after repeated bounded probes	Low detectability and weak incident response	Security telemetry, prompt-level canaries, SIEM correlation, response playbooks

5. RISK SCORING MODEL

A. Finding-Level Score

For each finding f, D-RELLM assigns normalized components in [0,1]: exploitability E_f, impact I_f, exposure X_f, autonomy/agency A_f, control bypass B_f, inverse detectability D_f, and mitigation strength M_f. The component vector is

$$x_f = [E_f, I_f, X_f, A_f, B_f, D_f, M_f]^T. \quad (1)$$

The finding score is

$$S_f = 10 \cdot clip_{(0,1)}(0.18E_f + 0.24I_f + 0.16X_f + 0.14A_f + 0.12B_f + 0.08D_f - 0.12M_f). \quad (2)$$

The weights emphasize impact and exploitability, add explicit credit for tool agency and exposure, and reduce score only for mitigations that have been replayed. Organizations can tune weights, but the weights should be fixed before testing so that scores are not adjusted after seeing results.

B. Confidence and Reproducibility

A black-box result should be discounted when it is ambiguous or difficult to reproduce. Let n_f be successful reproductions, a_f total bounded attempts, q_f evidence quality, v_f version stability, and s_f semantic ambiguity. We define confidence as

$$\kappa_f = ((n_f + 1)/(a_f + 2))\, q_f v_f (1 - s_f), \quad 0 \le \kappa_f \le 1. \quad (3)$$

The Laplace-smoothed replay ratio prevents one-off observations from dominating. Evidence quality is high when the outcome is captured with transcript hashes, timestamps, canary values, and screenshots; version stability is low when model or policy versions change during testing; semantic ambiguity is high when the output might have a benign explanation.

C. Application-Level Risk

The application score aggregates the top k de-duplicated findings through a bounded union model:

$$R_{app} = 100\,(1 - \prod^{k}_{f=1} (1 - \kappa_f S_f/10)^{w_f}). \quad (4)$$

where w_f is an asset-criticality weight. The model avoids simply adding scores beyond 100 while still recognizing that several medium findings can create high aggregate risk. For trust-boundary paths in Fig. 2, path risk can be approximated as

$$P(\pi) = V_\pi \cdot \prod_{e \in \pi} T_e, \quad (5)$$

where T_e is the estimated trust-crossing weakness for edge e and V_π is the value of the asset or action reached by path π.

D. Mitigation Effectiveness

Mitigation is measured by replay, not by design intent. Let R_0 be the pre-control risk and R_1 the post-control risk under the same probe budget. The normalized effectiveness is

$$\eta = (R_0 - R_1)/\max(R_0, \varepsilon), \quad 0 \le \eta \le 1, \quad (6)$$

where ε prevents division by zero. A control is considered validated only when it reduces both the score and the confidence-adjusted reproducibility of the finding.

TABLE III

COMPONENT RUBRIC FOR INDIVIDUAL FINDINGS

Component	Practical scoring question

E	Can a normal user reproduce it reliably with few bounded attempts?
I	Could the outcome affect confidentiality, integrity, availability, safety, compliance, or money?
X	What sensitive data, tenants, tools, identities, or downstream systems are reachable?
A	Can the application act, chain tools, persist memory, or propagate content without review?
B	Did the result bypass stated policy, role, ACL, confirmation, schema, or citation controls?
D	Would routine monitoring detect both the attempt and the outcome? Higher means less detectable.
M	Are tested mitigations layered, least-privilege, and effective against replay variants?

6. IMPLEMENTATION PROCEDURE

The procedure below is suitable for an internal assessment runbook. It is written as a defensive process rather than an offensive payload library.

Two details matter in practice. First, black-box tests should be randomized enough to avoid overfitting to one prompt but stable enough to support reproduction. Second, assessors should separate application failures from policy disagreements.

TABLE IV

OPERATIONAL PROCEDURE FOR D-RELLM ASSESSMENT

Step	Action
1	Freeze scope, roles, test accounts, data boundaries, rate limits, and stop conditions.
2	Map observable surfaces: text, uploads, URLs, citations, memory, APIs, exports, and tool traces.
3	Build a preliminary trust graph and mark untrusted-to-privileged paths.
4	Select probe families from Table II and assign benign canaries.
5	Execute bounded probes with randomization, replay limits, and no production secrets.
6	Cluster outcomes into findings and remove duplicates caused by the same root behavior.
7	Score each finding using Eq. (2) and confidence using Eq. (3).
8	Map mitigations, replay variants, compute residual risk, and document evidence hashes.

A refusal that is terse but safe is not a vulnerability; a fluent answer that violates an authorization boundary is.

7. ILLUSTRATIVE PILOT CASE STUDY

A. Synthetic Harnesses

We evaluated the framework on three controlled archetypes implemented as internal harnesses with synthetic data. The first was a customer-service chatbot with no external tools. The second was a RAG knowledge assistant with a seeded corpus, per-document metadata, and citation output. The third was a tool-using operations agent with dry-run functions for ticket creation, calendar updates, and file labeling. The values are illustrative of the scoring method and are not claims about any commercial product.

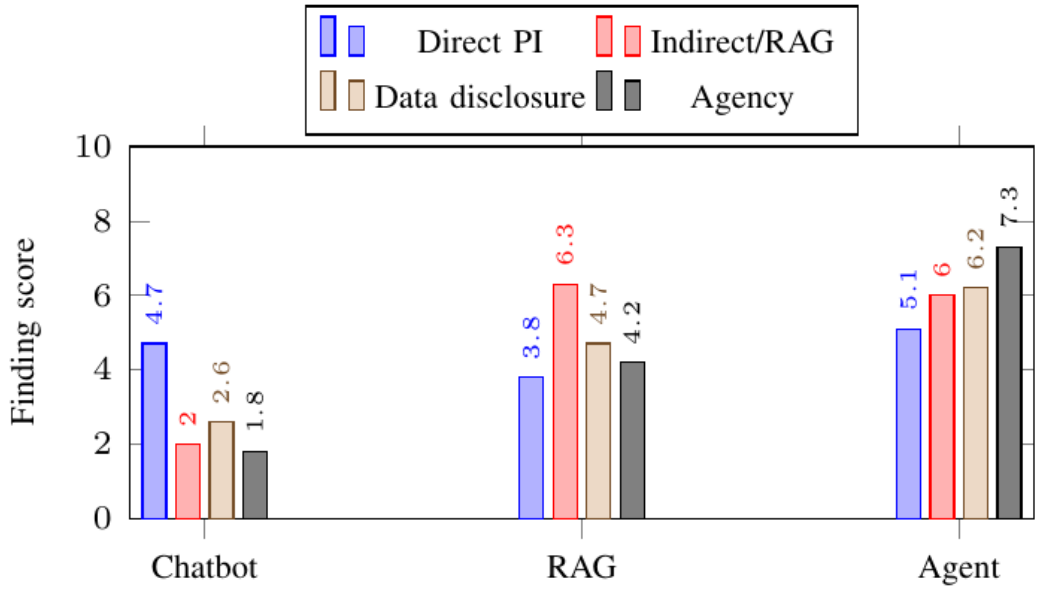

Fig. 3. Illustrative D-RELLM scores for synthetic application archetypes. Scores rank remediation urgency and are not measurements of real products.

B. Results

Table V shows representative component scores. The chatbot's dominant risk was direct instruction confusion. The RAG assistant had higher exposure and integrity risk because retrieved content could influence answers. The tool agent produced the highest risk because successful instruction confusion could reach action paths, even though all actions were sandboxed.

TABLE V

ILLUSTRATIVE SCORING RESULTS ON SYNTHETIC LLM APPLICATION ARCHETYPES

Archetype	Finding	E	I	X	A	B	D	M	S_f	Interpretation
Chatbot	Direct prompt injection	.70	.55	.20	.10	.55	.45	.20	4.7	User-channel conflict changed style and role behavior, but no private data or tools were reachable.
Chatbot	Output handling	.45	.50	.20	.10	.30	.50	.25	3.3	Structured output could

										confuse a downstream renderer if copied without encoding.
RAG assistant	Indirect injection	.65	.70	.55	.35	.70	.55	.15	6.3	Sandbox document content influenced task behavior beyond summarization.
RAG assistant	RAG integrity	.60	.68	.50	.25	.55	.60	.20	5.5	Citation dominance and stale-source selection reduced answer integrity.
Tool agent	Excessive agency	.55	.85	.70	.85	.75	.65	.10	7.3	Dry-run write action was prepared without sufficient confirmation.
Tool agent	Sensitive data disclosure	.45	.80	.75	.45	.60	.60	.15	6.2	Synthetic cross-context memory marker appeared in the wrong session.

C. Mitigation Replay

After initial scoring, three controls produced the largest reduction: strict separation of trusted instructions from untrusted data, least-privilege tool tokens with explicit human confirmation for write-like actions, and evidence-rich monitoring that linked prompts, tool calls, and canary events. Table VI shows example residual-risk changes under the same bounded replay budget.

TABLE VI

EXAMPLE MITIGATION REPLAY SUMMARY

Finding class	**Before**	**After**	**η**	**Main control credited**
Indirect injection	6.3	2.4	.62	Data/instruction separation and retrieval quarantine.
Excessive agency	7.3	2.1	.71	Human confirmation and scoped dry-run tokens.

Data disclosure	6.2	2.7	.56	ACL-preserving retrieval and memory isolation.
Output handling	3.3	1.4	.58	Typed schema and output encoding.

8. MITIGATION PLAYBOOK

A mitigation should reduce the opportunity for instruction confusion, reduce the value reachable through a compromised path, or increase detection and containment. Table VII maps common findings to controls and residual-risk tests. The important principle is that controls are credited only after replay. A policy document, banner, or static warning does not reduce risk unless the application behavior changes under test.

TABLE VII

MITIGATION MAPPING FOR D-RELLM FINDINGS

Finding	Primary controls	Residual-risk test
Prompt hierarchy failure	Model instruction hierarchy, structured query formats, policy-specific fine-tuning, refusal examples	Verify that lower-trust user/tool text cannot override developer policy in paraphrased variants.
RAG poisoning exposure	Corpus trust tiers, signed source ingestion, ACL-preserving retrieval, source diversity, citation verification	Inject benign canary documents and confirm they cannot dominate unrelated answers.
Tool overreach	Least-privilege OAuth scopes, dry-run mode, confirmation gates, transaction caps, reversible execution	Confirm high-impact tools require human approval and log purpose, identity, and arguments.
Prompt/system leakage	Keep secrets out of prompts, split policies from runtime secrets, use canaries, redact tool schemas	Confirm the app refuses to reveal hidden instructions while still explaining user-facing behavior.
Unsafe downstream output	Typed schemas, output encoding, allowlisted renderers, sandboxed code execution	Feed outputs into downstream parsers and verify no command, script, formula, or SQL interpretation occurs.
Unbounded consumption	Context limits, rate limits, loop counters, budget-aware planning, anomaly detection	Attempt benign loops and verify deterministic stop conditions and alerts.
Monitoring gap	Prompt-level canaries, trace IDs, tool-call audit events, SIEM correlation, incident runbooks	Confirm repeated bounded probes generate triage-ready events without logging sensitive prompt content.

9. REPORTING AND OPERATIONAL INTEGRATION

A D-RELLM report should be understandable to both security engineers and application owners. Each finding should include a plain-language summary, affected surface, business impact, evidence record, score components, confidence, mitigation owner, and replay status. The most useful reports avoid sensational jailbreak language and instead state what control failed. For example, “RAG content can override task instructions in a sandbox corpus” is more actionable than “the model was jailbroken.”

In engineering workflows, D-RELLM can be integrated at three gates. During design review, the trust graph identifies risky paths before implementation. During pre-release testing, bounded probes become regression tests for prompts, retrievers, and tools. During production monitoring, the evidence schema informs alert design: suspicious prompt patterns alone are not enough; alerts should connect user identity, retrieved documents, tool calls, output schema violations, and canary events.

Risk scoring should also support exception handling. Some applications intentionally have broad capabilities, such as an internal coding assistant or operations copilot. In those cases the right question is not whether the model can produce powerful output, but whether the application constrains identity, scope, approval, data access, and downstream execution. D-RELLM makes this explicit by scoring exposure, agency, bypass, and detectability separately.

10. LIMITATIONS AND FUTURE WORK

Black-box assessment cannot prove absence of vulnerabilities. It can miss rare failures, tenant-specific configuration bugs, model-version regressions, privileged workflows, and vulnerabilities reachable only through source-code or log inspection. It can also overstate risk if a surprising response is not reproducible or if the apparent behavior is caused by a benign product feature. That is why confidence and replay are central to D-RELLM.

The scoring weights in Eq. (2) are transparent but not universal. Financial institutions, healthcare systems, developer tools, and consumer chatbots may need different impact weights. Future work should calibrate weights against incident datasets, create inter-rater reliability studies, standardize machine-readable finding exchange, and connect black-box evidence to white-box prompt and code review. Another useful extension is continuous evaluation: a CI job could replay safe canary probes whenever a prompt template, retriever, model, or tool policy changes.

11. CONCLUSION

Defensive reverse engineering of LLM applications is becoming a necessary security practice because many deployed systems expose only behavior while hiding prompts, retrieval logic, tools, memory, and model choices. D-RELLM provides a safe and auditable black-box workflow for turning that behavior into risk decisions. By combining surface mapping, trust-boundary graphs, bounded probes, evidence-backed scoring, confidence adjustment, and mitigation replay, defenders can prioritize the failures that matter most: instruction confusion, sensitive-data leakage, RAG integrity failure, excessive agency, unsafe downstream output, and unbounded consumption. The central lesson is that LLM security should be measured at the application boundary, where models, data, tools, identity, and operations meet.

REFERENCES

1. OWASP GenAI Security Project, “OWASP Top 10 for Large Language Model Applications,” Open Worldwide Application Security Project, Tech. Rep., 2025. [Online]. Available: https://owasp.org/ www-project-top- 10-for-large-language-model-applications/
2. National Institute of Standards and Technology, “Artificial Intelligence Risk Management Framework (AI RMF 1.0),” NIST, Tech. Rep. NIST AI 100-1, 2023.
3. “Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile,” NIST, Tech. Rep. NIST AI 600-1, 2024.
4. “Secure Software Development Framework (SSDF) Version 1.1,” NIST, Tech. Rep. NIST SP 800-218, 2022.
5. R. Ross, “Guide for Conducting Risk Assessments,” NIST, Tech. Rep. NIST SP 800-30 Revision 1, 2012.
6. Forum of Incident Response and Security Teams, “Common Vulner-ability Scoring System Version 4.0: Specification Document,” 2023. [Online]. Available: https://www.first.org/cvss/specification- document
7. OWASP Foundation, “OWASP Risk Rating Methodology,” 2024. [Online]. Available: _https://owasp.org/www-community/OWASP_ Risk_Rating_Methodology
8. The Open Group, “Factor Analysis of Information Risk (FAIR) Standard, Version 3.0,” The Open Group, 2025. [Online]. Available: https://www.opengroup.org/open-fair
9. MITRE Corporation, “MITRE ATLAS: Adversarial Threat Landscape for Artificial-Intelligence Systems,” 2025. [Online]. Available: https: //atlas.mitre.org/
10. “MITRE ATTACK,” 2025. [Online]. Available: https://attack. mitre.org/

11. International Organization for Standardization, “ISO/IEC 23894:2023 Artificial Intelligence - Guidance on Risk Management,” ISO/IEC, 2023.
12. “ISO/IEC 42001:2023 Artificial Intelligence Management Sys-tem,” ISO/IEC, 2023.
13. UK National Cyber Security Centre, “Prompt Injection is Not SQL Injection (It May Be Worse),” 2025. [Online]. Available: https: //www.ncsc.gov.uk/blog- post/prompt-injection-is-not-sql-injection
14. Y. Liu, G. Deng, Y. Li, K. Wang, Z. Wang, X. Wang, T. Zhang, Y. Liu H. Wang, Y. Zheng, and Y. Liu, “Prompt Injection Attack against LLM: integrated Applications,” 2023, arXiv:2306.05499.
15. K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and, M. Fritz, “Not What You’ve Signed Up For: Compromising Real: World LLM-Integrated Applications with Indirect Prompt Injection, 2023, arXiv:2302.12173.
16. Y. Liu, Y. Jia, R. Geng, J. Jia, and N. Z. Gong, “Formalizing and Benchmarking Prompt Injection Attacks and Defenses,” in 33rd USENIX Security Symposium, 2024.
17. S. Chen, J. Piet, C. Sitawarin, and D. Wagner, “StruQ: Defending Against Prompt Injection with Structured Queries,” in 34th USENIX Security Symposium, 2025, arXiv:2402.06363.
18. E. Wallace, K. Xiao, R. Leike, L. Weng, J. Heidecke, and A. Beutel “The Instruction Hierarchy: Training LLMs to Prioritize Privileged Instructions,” 2024, arXiv:2404.13208.
19. Q. Zhan, Z. Liang, Z. Ying, and D. Kang, “InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents,” in Findings of the Association for Computational Linguistics. ACL 2024, 2024, pp. 10471-10506.
20. L. Derezynski, E. Galinkin, J. Martin, and S. Majumdar, “garak A Framework for Security Probing Large Language Models,” 2024 arXiv:2406.11036.
21. Microsoft AI Red Team, “PyRIT: Python Risk Identification Tool for Generative AI,” 2024. [Online]. Available: https://github.com/ microsoft/pyrit
22. M. Bhatt, S. Chennabasappa, Y. Li, C. Nikolaidis, D. Song, S. Wan F. Ahmad, C. Aschermann, Y. Chen, D. Kapil, D. Molnar, S. Whitman and J. Saxe, “CyberSecEval 2: A Wide-Ranging Cybersecurity Evalu: ation Suite for Large Language Models,” 2024, arXiv:2404.13161.
23. H. Pearce, B. Ahmad, B. Tan, B. Dolan-Gavitt, and R. Karri, “Purple Llama CyberSecEval: A Secure Coding Benchmark for Language Models,” 2023, arXiv:2312.04724.
24. N. Carlini, F. Tramér, E. Wallace, M. Jagielski, A. Herbert-Voss K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, A. Oprea, and, C. Raffel, “Extracting Training Data from Large Language Models, in 30th USENIX Security Symposium, 2021.
25. N. Carlini, C. Liu, U. Erlingsson, J. Kos, and D. Song, “The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks,” in 28th USENIX Security Symposium, 2019.
26. N. Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramér, and C. Zhang “Quantifying Memorization Across Neural Language Models,” in In: ternational Conference on Learning Representations, 2023.
27. Z. Deng, Y. Guo, C. Han, W. Ma, J. Xiong, S. Wen, and Y. Xiang “AI Agents Under Threat: A Survey of Key Security Challenges and, Future Pathways,” ACM Computing Surveys, 2025, arXiv:2406.02630
28. H. Zhang, J. Huang, K. Mei, Y. Yao, Z. Wang, C. Zhan, H. Wang and Y. Zhang, “Agent Security Bench (ASB): Formalizing and Bench: marking Attacks and Defenses in LLM-based Agents,” in International Conference on Learning Representations, 2025.
29. S. Schulhoff, J. Pinto, A. Khan, L.-F. Bouchard, C. Si, S. Anati V. Tagliabue, A. L. Kost, C. Carnahan, and J. Boyd-Graber, “Ignore This Title and HackAPrompt: Exposing Systemic Vulnerabilities of LLMs through a Global Scale Prompt Hacking Competition,” in Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, 2023.
30. S. Toyer, O. Watkins, E. A. Mendes, J. Svegliato, L. Bailey, T. Wang, I. Ong, K. Elmaaroufi, P. Abbeel, and S. Russell, “Tensor Trust: Interpretable Prompt Injection Attacks from an Online Game,” 2023, arXiv:2311.01011.
31. A. Zou, Z. Wang, N. Carlini, M. Nasr, J. Z. Kolter, and M. Fredrikson, “Universal and Transferable Adversarial Attacks on Aligned Language Models,” 2023, arXiv:2307.15043.
32. A. Wei, N. Haghtalab, and J. Steinhardt, “Jailbroken: How Does LLM Safety Training Fail?” in Advances in Neural Information Processing Systems, 2023.
33. P. Chao, A. Robey, E. Dobriban, H. Hassani, G. J. Pappas, and, E. Wong, “Jailbreaking Black Box Large Language Models in Twenty Queries,” in NeurIPS Workshop on Robustness of Few-shot and Zero-shot Learning in Foundation Models, 2023, arXiv:2310.08419.
34. A. Mehrotra, M. Zampetakis, P. Kassianik, B. Nelson, H. Anderson, Y. Singer, and A. Karbasi, “Tree of Attacks: Jailbreaking Black-Box LLMs Automatically,” 2023, arXiv:2312.02119.
35. J. Yu, X. Lin, Z. Yu, and X. Xing, “GPTFUZZER: Red Teaming Large Language Models with Auto-Generated Jailbreak Prompts,” 2023, arXiv:2309.10253.
36. X. Liu, N. Xu, M. Chen, and C. Xiao, “AutoDAN: Generat-ing Stealthy Jailbreak Prompts on Aligned Large Language Mod-els,” in International Conference on Learning Representations, 2024, arXiv:2310.04451.
37. G. Deng, Y. Liu, Y. Li, K. Wang, Y. Zhang, Z. Li, H. Wang, T. Zhang, and Y. Liu, “MasterKey: Automated Jailbreak Across Multiple Large Language Model Chatbots,” in Network and Distributed System Secu-rity Symposium, 2024, arXiv:2307.08715.

38. K. Zhu, J. Wang, J. Zhou, Z. Wang, H. Chen, Y. Wang, L. Yang, W. Ye, Y. Zhang, N. Z. Gong, and X. Xie, "PromptBench: Towards Evaluating the Robustness of Large Language Models on Adversarial Prompts," 2023, arXiv:2306.04528.
39. E. Perez, S. Huang, F. Song, T. Cai, R. Ring, J. Aslanides, A. Glaese, N. McAleese, and G. Irving, "Red Teaming Language Models with Language Models," in Proceedings of the 2022 Conference on Empir-ical Methods in Natural Language Processing, 2022.
40. D. Ganguli, L. Lovitt, J. Kernion, A. Askell, Y. Bai, S. Kadavath, B. Mann, E. Perez, N. Schiefer et al., "Red Teaming Language Models to Reduce Harms: Methods, Scaling Behaviors, and Lessons Learned," 2022, arXiv:2209.07858.
41. P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Kuttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and, D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Advances in Neural Information Processing Systems, 2020.
42. V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. tau Yih, "Dense Passage Retrieval for Open-Domain Question Answering," in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, 2020.
43. K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "REALM: Retrieval-Augmented Language Model Pre-Training," in Proceedings of the 37th International Conference on Machine Learning, 2020.
44. G. Izacard and E. Grave, "Leveraging Passage Retrieval with Gener-ative Models for Open Domain Question Answering," in Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, 2021.
45. N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych "BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation ot Information Retrieval Models," in Advances in Neural Informatior Processing Systems Datasets and Benchmarks Track, 2021.
46. N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embedding: using Siamese BERT-Networks," in Proceedings of the 2019 Confer: ence on Empirical Methods in Natural Language Processing, 2019.
47. J. Johnson, M. Douze, and H. Jégou, "Billion-scale Similarity Searct with GPUs," IEEE Transactions on Big Data, vol. 7, no. 3, pp. 535-547, 2021, arXiv:1702.08734.
48. W. Zou, R. Geng, B. Wang, and J. Jia, "PoisonedRAG: Knowl. edge Corruption Attacks to Retrieval-Augmented Generation of Largȼ Language Models," in 34th USENIX Security Symposium, 2025 arXiv:2402.07867.
49. Q. Zhang, B. Zeng, C. Zhou, G. Go, H. Shi, and Y. Jiang, "Human: Imperceptible Retrieval Poisoning Attacks in LLM-Powered Appi: cations," in Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering, 2024, pp 502-506, arXiv:2404.17196.
50. A. RoyChowdhury, M. Luo, P. Sahu, S. Banerjee, and M. Tiwari "ConfusedPilot: Confused Deputy Risks in RAG-based LLMs," 2024 arXiv:2408.04870.
51. C. Clop and Y. Teglia, "Backdoored Retrievers for Prompt Injection At tacks on Retrieval Augmented Generation of Large Language Models, 2024, arXiv:2410.14479.
52. T. Schick, J. Dwivedi-Yu, R. Dessi, R. Raileanu, M. Lomeli, L. Zettle-moyer, N. Cancedda, and T. Scialom, "Toolformer: Language Model: Can Teach Themselves to Use Tools," in Advances in Neural Informa: tion Processing Systems, 2023.
53. S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao "ReAct: Synergizing Reasoning and Acting in Language Models," in International Conference on Learning Representations, 2023.
54. E. Karpas, O. Abend, Y. Bisk, S. Shalev-Shwartz, Y. Shoham, and, A. Shashua, "MRKL Systems: A Modular, Neuro-Symbolic Archi: tecture that Combines Large Language Models, External Knowledge Sources and Discrete Reasoning," 2022, arXiv:2205.00445.
55. Y. Qin, S. Hu, Y. Lin, W. Chen, N. Ding, G. Cui, Z. Zeng, Y. Huang C. Xiao, C. Han et al., "ToolLLM: Facilitating Large Language Model: to Master 16000+ Real-world APIs," 2023, arXiv:2307.16789.
56. S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez, "Gorilla: Larg« Language Model Connected with Massive APIs," in Advances in Neural Information Processing Systems, 2023, arXiv:2305.15334.
57. Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, "HuggingGPT Solving AI Tasks with ChatGPT and its Friends in Hugging Face, 2023, arXiv:2303.17580.
58. R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse S. Jain, V. Kosaraju, W. Saunders et al., "WebGPT: Browser-assisted Question-answering with Human Feedback," 2021, arXiv:2112.09332
59. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," in Advances in Neural Information Processing Systems, 2017.
60. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understand-ing," in Proceedings of NAACL-HLT, 2019.
61. T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell er al., "Language Models are Few-Shot Learners," Advances in Neural Information Processing Systems, 2020.
62. C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," Journal of Machine Learning Research, vol. 21, no. 140, pp. 1-67, 2020.
63. L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray et al., "Training Language Models to Follow Instructions with Human Feedback," in Advances in Neural Information Processing Systems, 2022.

64. Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan er al., "Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback," 2022, arXiv:2204.05862.
65. Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon et al., "Constitutional AI: Harmlessness from AI Feedback," 2022, arXiv:2212.08073.
66. OpenAlI, "GPT-4 Technical Report," 2024, arXiv:2303.08774.
67. H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Roziére, N. Goyal, E. Hambro, F. Azhar et al., "LLaMA: Open and Efficient Foundation Language Models," 2023, arXiv:2302.13971.
68. H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale ef al., "Llama 2: Open Foundation and Fine-Tuned Chat Models," 2023, arXiv:2307.09288.
69. Gemini Team, "Gemini: A Family of Highly Capable Multimodal Models," 2023, arXiv:2312.11805.
70. P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar et al., "Holistic Evaluation of Language Models," 2022, arXiv:2211.09110.
71. D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and, J. Steinhardt, "Measuring Massive Multitask Language Understanding," in International Conference on Learning Representations, 2021.
72. S. Lin, J. Hilton, and O. Evans, "TruthfulQA: Measuring How Models Mimic Human Falsehoods," in Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, 2022.
73. A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso et al., "Beyond the Imitation Game: Quantifying and Extrapolating the Capabilities of Language Models," in Transactions on Machine Learning Research, 2022.
74. S. Gehman, S. Gururangan, M. Sap, Y. Choi, and N. A. Smith, "Real-ToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models," in Findings of the Association for Computational Linguistics: EMNLP 2020, 2020.
75. T. Hartvigsen, S. Gabriel, H. Palangi, M. Sap, D. Ray, and E. Kamar, "ToxiGen: A Large-Scale Machine-Generated Dataset for Adversarial and Implicit Hate Speech Detection," in Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, 2022.
76. A. Parrish, A. Chen, N. Nangia, V. Padmakumar, J. Phang, J. Thomp-son, P. M. Htut, and S. Bowman, "BBQ: A Hand-Built Bias Benchmark for Question Answering," in Findings of the Association for Computa-tional Linguistics: ACL 2022, 2022.
77. N. Nangia, C. Vania, R. Bhalerao, and S. R. Bowman, "CrowS-Pairs: A Challenge Dataset for Measuring Social Biases in Masked Language Models," in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, 2020.
78. M. Nadeem, A. Bethke, and S. Reddy, "StereoSet: Measuring Stereo-typical Bias in Pretrained Language Models," in Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics, 2021.
79. M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh, "Beyond Accuracy: Behavioral Testing of NLP Models with CheckList," in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020.
80. A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding," in International Conference on Learning Representations, 2019.
81. A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill O. Levy, and S. R. Bowman, "SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems," in Advances in Neural Information Processing Systems, 2019.
82. P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text," in Proceedings 0, the 2016 Conference on Empirical Methods in Natural Language Processing, 2016.
83. P. Rajpurkar, R. Jia, and P. Liang, "Know What You Don't Know Unanswerable Questions for SQuAD," in Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics 2018.
84. R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership Inference Attacks Against Machine Learning Models," in 2017 IEEE Symposium on Security and Privacy, 2017, pp. 3-18.
85. F. Tramér, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing Machine Learning Models via Prediction APIs," in 25th USENIX Security Symposium, 2016.
86. T. Orekondy, B. Schiele, and M. Fritz, "Knockoff Nets: Stealing Functionality of Black-Box Models," in IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019.
87. N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and, A. Swami, "Practical Black-Box Attacks against Machine Learning, in Proceedings of the 2017 ACM Asia Conference on Computer and, Communications Security, 2017.
88. I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harness-ing Adversarial Examples," in International Conference on Learning Representations, 2015.
89. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Good fellow, and R. Fergus, "Intriguing Properties of Neural Networks," in International Conference on Learning Representations, 2014.
90. N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," in 2017 IEEE Symposium on Security and Privacy 2017, pp. 39-57.
91. A. Athalye, N. Carlini, and D. Wagner, "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples," in Proceedings of the 35th International Conference or Machine Learning, 2018.

92. A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards Deep Learning Models Resistant to Adversarial Attacks,” in Interna: tional Conference on Learning Representations, 2018.
93. J. Cohen, E. Rosenfeld, and Z. Kolter, “Certified Adversarial Ro-bustness via Randomized Smoothing,” in Proceedings of the 36th International Conference on Machine Learning, 2019.
94. A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry “Adversarial Examples Are Not Bugs, They Are Features,” in Advances in Neural Information Processing Systems, 2019.
95. B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Smdié, P. Laskov G. Giacinto, and F. Roli, “Evasion Attacks against Machine Learning at Test Time,” in Joint European Conference on Machine Learning and, Knowledge Discovery in Databases, 2013.
96. M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, “The Security of Machine Learning,” Machine Learning, vol. 81, no. 2, pp. 121-148 2010.
97. E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh, “Uni-versal Adversarial Triggers for Attacking and Analyzing NLP,” in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, 2019.
98. J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, “HotFlip: White-Box Adversarial Examples for Text Classification,” in Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics 2018.
99. D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, “Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment,” in Proceedings of the AAAI Conference on Artificial Intelligence, 2020.
100. J. Li, S. Ji, T. Du, B. Li, and T. Wang, “TextBugger: Generating Adversarial Text Against Real-world Applications,” in Network and, Distributed System Security Symposium, 2019.
101. L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu, “BERT-ATTACK: Adver-sarial Attack Against BERT Using BERT,” in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, 2020.
102. R. Jia and P. Liang, “Adversarial Examples for Evaluating Reading Comprehension Systems,” in Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017.
103. J. X. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, and Y. Qi, “TextAttack: A Framework for Adversarial Attacks, Data Augmenta-tion, and Adversarial Training in NLP,” in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2020.
104. H. Inan, K. Upasani, J. Chi, R. Rungta, K. Iyer, Y. Mao, M. Tontchev, Q. Hu, B. Fuller, D. Testuggine, and M. Khabsa, “Llama Guard: LLM-based Input-Output Safeguard for Human-AI Conversations,” 2023, arXiv:2312.06674.
105. T. Rebedea, R. Dinu, M. N. Sreedhar, C. Parisien, and J. Cohen, “NeMo Guardrails: A Toolkit for Controllable and Safe LLM Applications with Programmable Rails,” in Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demon-strations, 2023.
106. Guardrails AI, “Guardrails Documentation,” 2026. [Online]. Available: https://guardrailsai.com/guardrails/docs
107. LangChain, “Guardrails for Agents,’ 2026. [Online]. Available: https://docs.langchain.com/oss/python/langchain/guardrails
108. M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchin-son, E. Spitzer, I. D. Raji, and T. Gebru, “Model Cards for Model Reporting,” in Proceedings of the Conference on Fairness, Account-ability, and Transparency, 2019.
109. T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. D. III, and K. Crawford, “Datasheets for Datasets,” Communications of the ACM, vol. 64, no. 12, pp. 86-92, 2021.
110. I. D. Raji, A. Smart, R. N. White, M. Mitchell, T. Gebru, B. Hutchin-son, J. Smith-Loud, D. Theron, and P. Barnes, “Closing the AI Accountability Gap: Defining an End-to-End Framework for Internal Algorithmic Auditing,” in Proceedings of the 2020 Conference of Fairness, Accountability, and Transparency, 2020.
111. E. Strubell, A. Ganesh, and A. McCallum, “Energy and Policy Consid-erations for Deep Learning in NLP,” in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019.
112. R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. vor Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill et al., “On the Opportunities and Risks of Foundation Models,” in arXiv preprint arXiv:2108.07258, 2021.
113. E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell “On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?” in Proceedings of the 2021 ACM Conference on Fairness. Accountability, and Transparency, 2021.
114. L. Weidinger, J. Mellor, M. Rauh, C. Griffin, J. Uesato, P.-S. Huang M. Cheng, M. Glaese, B. Balle, A. Kasirzadeh et al., “Taxonomy of Risks Posed by Language Models,” Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency, 2022.
115. Y. Yao, J. Duan, K. Xu, Y. Cai, Z. Sun, and Y. Zhang, “A Survey on Large Language Model (LLM) Security and Privacy: The Good the Bad, and the Ugly,” High-Confidence Computing, vol. 4, no. 2, p 100211, 2024.
116. X. Huang, W. Ruan, W. Huang, G. Jin, Y. Dong, C. Wu, S. Bensalem R. Mu, Y. Qi, X. Zhao et al., “A Survey of Safety and Trustworthiness of Large Language Models through the Lens of Verification and, Validation,” Artificial Intelligence Review, vol. 57, 2024.
117. Y. Liu, Y. Jia, J. Jia, and N. Z. Gong, “Security and Privacy Challenges of Large Language Models: A Survey,” ACM Computing Surveys, 2025
118. Meta AI, “Purple Llama: Open Trust and Safety Tools for Generative AI,” 2023. [Online]. Available: https://ai.meta.com/blog, purple-llama-open-trust-safety-generative-ai/
119. OpenAI, “Model Spec,” 2025. [Online]. Available: https://model-spec openai.com/
120. Anthropic, “Constitutional Classifiers: Defending Against Universal Jailbreaks,” 2025. [Online]. Available: https://www.anthropic.com research/constitutional-classifiers