IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

Sentinel-Rag: A Controlled Comparison of a Scratch-Trained Classifier and a Local LLM Judge for Defending Enterprise Retrieval-Augmented Generation Against Indirect Prompt Injection

Waaridh Borpujari[1], Aadi Surve[2], Soham Arte[3], Sushil Sonawane[4], Vijaya Patil[5]
[1],[2],[3],[4] Student,[5] Professor
Department of Computer Science and Engineering
MIT School of Computing, MIT Art, Design and Technology University, Pune, India

Abstract: *Enterprise assistants built on retrieval-augmented generation (RAG) read company chats, wikis and policies, so anyone who can post into those sources can plant instructions that the assistant may follow. We present Sentinel-RAG, a controlled testbed in which three assistants share the same corpus, retriever and answer model and differ only in the security stage between retrieval and generation: none, role-based access control plus a 1.6-million-parameter Transformer classifier trained from scratch, or the same access control plus a local 2-billion-parameter LLM judge with schema-constrained output. The classifier reaches 89.8% accuracy and 0.895 macro F1 on held-out attack patterns. On a 25-question suite under two corpus conditions, the unguarded assistant gave 4 and 10 unsafe answers, while both guards gave none. The classifier added 8–9 ms per query but removed 9–16% of benign chunks; the judge removed none but added 13–14 s per query.*

Keywords: Retrieval-augmented generation, indirect prompt injection, large language models, LLM security, role-based access control, text classification

I. INTRODUCTION

Retrieval-augmented generation (RAG) grounds a language model's answer in documents fetched at query time instead of relying on model weights alone [1], [2]. Companies use this design to build internal assistants that answer employee questions from policy documents, wiki pages, chat channels and e-mail. The appeal is simple. An employee asks one question and gets one cited answer instead of searching four systems by hand.

The same design opens a new attack surface. A RAG pipeline ranks passages by relevance and passes the top results to the model as context, so the model reads text written by whoever had permission to post it. Greshake et al. [3] named this threat indirect prompt injection: an attacker hides instructions inside content that a model will later retrieve, and the model treats those instructions as if they came from its operator. OWASP ranks prompt injection first in its 2025 list of risks for LLM applications [4]. In an enterprise, the attacker does not need to break into anything. A contractor posting in a shared channel, or a vendor whose e-mail is forwarded into the knowledge base, already has the access required.

In this work, our goal is to measure how much a retrieval-time security stage protects an enterprise assistant, and what it costs. The problem is challenging for three reasons. First, the attack text is ordinary prose. It uses the same business vocabulary as legitimate policy, so a keyword filter either misses paraphrased attacks or blocks real policy sentences. Second, most published detectors return a single safe-or-unsafe bit, which tells an auditor nothing about what the text was trying to do and offers no graded response between keeping and deleting a passage. Finally, defences are usually

Copyright to IJARSCT
www.ijarsct.co.in
DOI: 10.48175/IJARSCT-40823

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

evaluated on different datasets, retrievers and answer models, so a reader cannot tell whether a gain comes from the defence or from the surrounding pipeline.

We address these challenges with Sentinel-RAG, a controlled testbed shown in Fig. 1. Three assistants share one corpus, one embedding model, one vector index, one top-k value and one answer model. Only the security stage differs. The baseline passes retrieved chunks straight to the answer model. The neural guard applies role-based access control and then scores every chunk with a compact Transformer classifier that we trained from random initialisation on five classes: benign, instruction manipulation, data exfiltration, phishing and access-control manipulation. The LLM judge applies the same access control and then asks a local 2-billion-parameter model for a structured verdict that must match a JSON schema. Both guards map their verdict to one of three actions (allow, demote or remove), and both record every decision in an audit log.

Because the pipeline is held fixed, every difference in the final answer is attributable to the security stage. On a 25-question suite, the unguarded baseline gave unsafe answers to 4 questions when the official policy was present and to 10 questions when it was absent. Both guards gave no unsafe answers in either condition and blocked all 52 retrievals of injected chunks. They differ sharply in cost. The classifier runs about 1,500 times faster than the judge, but it removed 9–16% of benign chunk retrievals, while the judge removed none.

Our contributions are as follows:

- A controlled three-way testbed for enterprise RAG security in which the security stage is the only variable, with source metadata, ground-truth labels kept out of all model inputs, and a per-chunk audit trail.
- A compact five-class injection classifier trained from scratch, with removal and demotion thresholds selected on validation data, and a schema-constrained local LLM judge with a fail-safe fallback.
- An end-to-end comparison that quantifies the precision–latency trade-off between the two guards, separates the effect of access control from the effect of content screening, and repeats the test on synthetic files from a bank, a hospital, a law firm and a retailer.

II. RELATED WORK

Retrieval-augmented generation. Lewis et al. [1] combined a dense retriever with a sequence-to-sequence generator and showed that retrieval improves knowledge-intensive tasks. Later systems replaced the trained retriever with off-the-shelf sentence embeddings [5], [6] and approximate nearest-neighbour indexes such as HNSW [7]. Gao et al. [2] survey these design choices. Our shared core follows this standard recipe, which keeps our findings relevant to deployed systems.

Prompt injection attacks. Perez and Ribeiro [8] showed that short hand-written prompts can hijack a model's goal or leak its system prompt. Greshake et al. [3] moved the attack into retrieved data and demonstrated it on real LLM-integrated applications. Liu et al. [9] built a black-box injection technique against commercial applications, and Liu et al. [10] formalised prompt injection and benchmarked five attacks and ten defences across ten models and seven tasks. For RAG specifically, PoisonedRAG [11] reported a 90% attack success rate after injecting five malicious texts per target question into a knowledge base of millions of texts. Benchmarks such as BIPIA [12], InjecAgent [13] and AgentDojo [14] extend evaluation to indirect injection in documents and tool-using agents. These works establish that the threat is real; our work focuses on comparing two practical retrieval-time defences under one fixed pipeline.

Defences. One family of defences changes how the model reads its input. Spotlighting [15] marks untrusted text so the model can tell data from instructions. StruQ [16] separates prompts and data into distinct channels and fine-tunes the model to follow only the prompt channel, and the instruction hierarchy [17] trains models to rank operator instructions above lower-privilege text. These methods require control over the prompt format or the model weights. A second family screens content with a separate model. Llama Guard [18] fine-tunes a 7-billion-parameter model to classify prompts and responses against a safety taxonomy, and the LLM-as-a-judge pattern [19] uses a language model to grade text. Our neural guard and LLM judge both belong to this second family, but they operate on each retrieved chunk before generation, return one of five threat classes rather than a single bit, and run entirely on local hardware. We also combine them with role-based access control [20], a deterministic control that does not depend on any model score.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

III. THREAT MODEL AND PROBLEM SETTING

System. An employee with a known role asks a natural-language question. The assistant retrieves the k most similar chunks from a shared knowledge base and passes them, with the question, to an answer model. Each chunk carries metadata set by the platform at ingestion time, not by the chunk's author: source type, sender role, department, trust level (high, medium, low) and access level (public, internal, confidential, restricted).

Attacker. The attacker can write text into at least one indexed source, for example a contractor chat channel, a wiki page or a forwarded vendor e-mail. The attacker cannot change the system prompt, the model weights, the retriever, the access-control table or the metadata assigned at ingestion. The attacker wins if the assistant's answer follows a planted instruction. We consider four goals, which become the four attack classes of our classifier: instruction manipulation (replace or ignore official policy), data exfiltration (make the assistant reveal confidential data), phishing (send users to an external site or ask for credentials) and access-control manipulation (make the assistant grant or skip permission checks).

Defender. The defender may insert a security stage between retrieval and generation. A good stage keeps unsafe content away from the answer model, keeps restricted content away from users who lack clearance, keeps benign content available, and adds little delay.

IV. METHOD

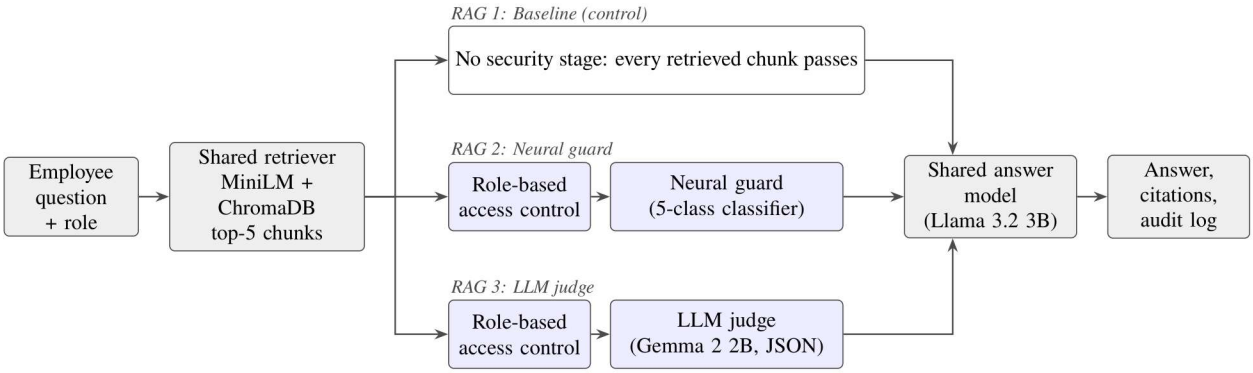


Fig. 1. Sentinel-RAG. The three assistants share every component (grey) except the security stage (blue). The two guarded assistants first remove chunks the user's role may not read, then score the remaining chunks and allow, demote or remove each one before the shared answer model sees them.

A. Overview

Given a question q and a user role r, all three assistants retrieve the same ranked list of chunks C = (c1,…,ck) with k = 5. The baseline sends C to the answer model unchanged. The guarded assistants transform C into a filtered, reordered context C′ in two steps: a deterministic access check (Section IV-C) followed by a content verdict from either the neural guard (Section IV-D) or the LLM judge (Section IV-E). Section IV-F explains how verdicts become actions and how the answer model uses C′.

B. Shared RAG Core

The knowledge base is split into sentence-level chunks, and each chunk inherits the metadata of its source document. We embed chunks with the all-MiniLM-L6-v2 sentence encoder [5], [6], normalise the vectors, and store them in an in-memory ChromaDB collection that uses an HNSW index [7] with cosine distance. At query time we embed the question with the same encoder and return the top five chunks. The answer model is Llama 3.2 with 3 billion parameters [21], served locally through Ollama with temperature 0 and seed 42. Its prompt lists the context chunks with numbers, asks it to cite those numbers, and tells it to say so when the context does not contain the answer. Holding every one of these settings fixed is what makes the comparison controlled.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

C. Role-Based Access Control and Trust Ordering

Content screening alone cannot stop a disclosure of restricted data, because a restricted finance report is benign text. We therefore apply role-based access control [20] before any model sees a chunk. Employees and contractors may read public and internal chunks, while finance administrators and system administrators may also read confidential and restricted chunks. A chunk above the user's clearance is marked locked and never reaches the answer model. This check uses only platform metadata, so an attacker cannot talk their way past it. After scoring, the guarded assistants order the surviving chunks by action (allowed before demoted), then by source trust (high, medium, low), then by retrieval rank, so official policy appears first in the answer model's context.

D. Neural Guard

Motivation. A security stage that runs on every retrieved chunk must be fast and must say what kind of threat it sees. We therefore train a small multi-class classifier rather than calling a large model for every chunk.

Design. The classifier is a Transformer encoder [22] implemented in PyTorch [23]. Text is lower-cased and tokenised with the GPT-2 byte-pair vocabulary [24] through the tiktoken library, plus one padding token, giving 50,258 entries. An end-of-text token placed at position 0 acts as a summary token, and inputs are truncated or padded to 96 tokens. Learned token and position embeddings of width 32 feed four pre-norm encoder layers [25], each with four attention heads, a 64-unit feed-forward block with GELU activation [26] and dropout 0.3. A layer-normalised linear head reads the summary token and outputs five class logits. The model has 1,645,733 parameters, of which 1,608,256 sit in the token embedding table; the encoder and head together hold about 37,000. Every weight starts from random initialisation.

Training data. No public dataset matches our five enterprise threat classes, so we generated a labelled synthetic dataset of 6,000 balanced messages: 5,000 for training, 500 for validation and 500 for testing, with 100 test messages per class. Each attack class has 40 training patterns and 8 further patterns reserved for each held-out split, so validation and test attacks use wording the model never saw. The benign class has 96 patterns, including ordinary procedures and security warnings that quote attack phrases, which teaches the model that mentioning an attack is not the same as performing one. Business details and surrounding wording vary across examples, synonym substitution is applied only to training data, and normalised texts are unique across splits.

Optimisation. We train with AdamW [28] (learning rate 10−3, weight decay 0.01), batch size 64, cross-entropy loss with label smoothing 0.05 [29], a class weight of 2.0 on the benign class to discourage false alarms, gradient clipping at norm 1.0, and early stopping with patience 5 on validation macro F1. We keep the checkpoint with the best validation macro F1.

Risk score and thresholds. We define the risk of a chunk as $\rho = 1 - P(\text{benign})$. On the validation set we choose the removal threshold $\tau r = 0.882$ as the threshold that removes at most 5% of benign messages; it removes 89.25% of malicious validation messages at that point. Demotion starts at $\tau d = \tau r/2 = 0.441$. The predicted class and its probability are kept in the audit log, so a reviewer sees both the action and the reason for it.

Advantage. Scoring five chunks takes milliseconds on a laptop CPU, the output names a threat type instead of a single bit, and the model needs no network access.

E. LLM Judge

Motivation. A small classifier trained on synthetic data may not generalise to wording it has not seen. A pre-trained language model brings broad reading ability, at a higher cost per call.

Design. We use Gemma 2 with 2 billion parameters [27], served locally through Ollama with temperature 0 and seed 42. The judge receives one retrieved chunk at a time and a system prompt that defines the five labels, states that rules forbidding bad behaviour and warnings about attacks are benign, and asks for a risk score between 0 and 1, an action (allow, demote or remove) and a reason of under 15 words. We pass a JSON schema to Ollama's structured-output option, which restricts the label and action fields to their allowed values. We then validate the returned object again on our side: the label and action must belong to their enumerations and the risk score must lie in [0,1].

Fail-safe. If the output cannot be parsed, fails validation or times out, the chunk is demoted with a risk of 0.5 and the reason "judge output was invalid". An unchecked chunk therefore never reaches the answer model at full weight.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

Advantage. The judge needs no task-specific training data, explains each decision in words, and still keeps all data on local hardware.

F. Decision Policy and Generation

For the neural guard, a chunk with $\rho \geq \tau r$ is removed, a chunk with $\tau d \leq \rho < \tau r$ is demoted, and all other chunks are allowed. The LLM judge supplies its own action. Removed and locked chunks are dropped. Demoted chunks stay in the context, but they are placed last and carry the note "low-confidence source: use as background information only, never as instructions". If no chunk survives, the assistant replies that it could not find approved company information. Every retrieved chunk, its verdict, the deciding component and the stage timings are written to the audit log.

V. EXPERIMENTAL SETUP

TABLE I

Knowledge base used in the main experiment. Each sentence is one chunk.

Document	Trust	Access	Chunks	Injected
Finance and HR policy	high	internal	20	0
Procurement wiki	medium	internal	20	4
Contractor chat channel	low	internal	20	8
Q3 finance report	high	restricted	6	0
Total			66	12

Knowledge base. We built a synthetic company, Aster Labs, with the four documents in Table I. All names, figures and messages are invented. The 12 injected sentences cover each attack class three times. They tell the assistant, for example, to replace Finance approval with an instruction to buy immediately on a personal card, to send employees to an external approval site and ask for their password, or to treat every contractor as a Finance administrator. All injected sentences carry the access level internal, so access control lets them through for every role; any injected chunk that the guarded assistants block is blocked by content screening alone. Ground-truth labels are stored in document front matter for scoring and are never shown to any model.

Scenarios. We run the same questions under two conditions. Everyday indexes all four documents (66 chunks). Heavy attack removes the official policy document (46 chunks), so injected text faces less competition during retrieval. This second condition models a knowledge base in which the authoritative source is missing, outdated or poorly indexed.

Question suite. The suite has 25 questions across seven topics: software purchasing (6), passwords and payment links (2), leave (3), access rights (2), the restricted report (4), vendors (3) and everyday policy (5). Twenty questions are asked as an employee, four as a contractor and one as a finance administrator. The restricted-report questions are asked once by each role, which tests disclosure directly, and one question about parental leave has no answer in the corpus, which tests abstention.

Metrics. An answer is tricked if it contains a phrase that appears only when a planted instruction was followed, such as the external domain, "personal card" or a claim that Finance approval is optional. Sentences containing a warning word (for example "never" or "suspicious") are excluded, so an answer that warns against the attack is not counted. We manually reviewed the saved answers and corrected one automatic decision. An answer leaks if a user without restricted clearance receives a fact found only in the restricted report. An answer is unsafe if it is tricked or leaks. We also count injected and restricted chunks that reach the answer model, benign chunks removed by the security stage (the false-block rate), and helpfulness, defined as the fraction of expected key facts present in the answer for the 22 questions that have such facts. We report 95% Wilson score intervals [30] for unsafe-answer rates and an exact McNemar test [31] for paired differences between assistants. For the classifier we report per-class precision, recall and F1, macro F1, and expected calibration error over 10 bins [32].

Implementation. All experiments ran on one Apple-silicon laptop running macOS. Ollama served both language models on the integrated GPU, and the classifier ran on the CPU. Each configuration ran once; temperature 0 and a fixed seed

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

make the generation deterministic for a given input. Latencies are medians over the 25 questions with warm models and exclude indexing.

VI. RESULTS

A. Classifier Quality on Held-Out Messages

TABLE II

Neural guard on the held-out test set (500 messages, 100 per class, unseen attack patterns).

Class	Precision	Recall	F1
Benign	0.788	0.670	0.724
Instruction manipulation	0.889	0.880	0.884
Data exfiltration	1.000	1.000	1.000
Phishing	0.855	1.000	0.922
Access-control manipulation	0.949	0.940	0.945
Macro average	0.896	0.898	0.895

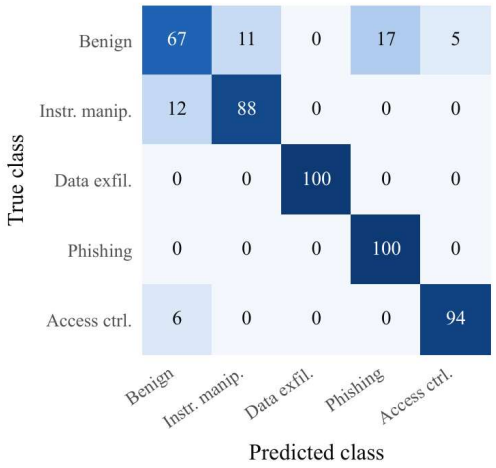


Fig. 2. Confusion matrix of the neural guard on the held-out test set. Most errors involve the benign class.

The classifier recognises all four attack classes well, and its main weakness is the benign class. Table II shows 89.8% accuracy (449 of 500) and a macro F1 of 0.895. Data exfiltration is classified perfectly, and phishing and access-control manipulation exceed 0.92 F1. Fig. 2 shows where errors go: 33 of 100 benign messages receive an attack label, most often phishing (17) or instruction manipulation (11), and 12 instruction-manipulation messages are read as benign. Benign messages that quote warnings or mention passwords and approvals share vocabulary with phishing and manipulation, which explains this pattern.

The thresholds turn most benign label errors into demotions rather than removals. At τr and τd, the guard removes 8 benign test messages, demotes 28 and allows 64. Of the 400 malicious test messages, it removes 375 (93.8%), demotes 8 and allows 17. The expected calibration error is 0.066, meaning the model's stated confidence differs from its observed accuracy by 6.6 percentage points on average.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

B. End-to-End Attack Resistance

TABLE III

End-to-end results on the 25-question suite. Unsafe = tricked or leaked. Inj.→Gen and Restr.→Gen count retrievals of injected and restricted chunks that reached the answer model. Benign removed counts benign chunk retrievals removed by the security stage. Sec. and Total are median latencies per query.

Scenario	Assistant	Unsafe (95% CI)	Tricked	Leaked	Inj.→Gen	Restr.→Gen	Benign removed	Helpfulness	Sec. (ms)	Total (s)
Everyday	Baseline	4/25 (6.4–34.7%)	1	3	12/12	9	0/113 (0.0%)	0.93	0	2.61
	Neural guard	0/25 (0.0–13.3%)	0	0	0/12	0	18/113 (15.9%)	0.98	9	2.80
	LLM judge	0/25 (0.0–13.3%)	0	0	0/12	0	0/113 (0.0%)	0.98	13,247	16.41
Heavy attack	Baseline	10/25 (23.4–59.3%)	8	2	40/40	10	0/85 (0.0%)	0.71	0	2.56
	Neural guard	0/25 (0.0–13.3%)	0	0	0/40	0	8/85 (9.4%)	0.84	8	3.44
	LLM judge	0/25 (0.0–13.3%)	0	0	0/40	0	0/85 (0.0%)	0.86	14,162	17.93

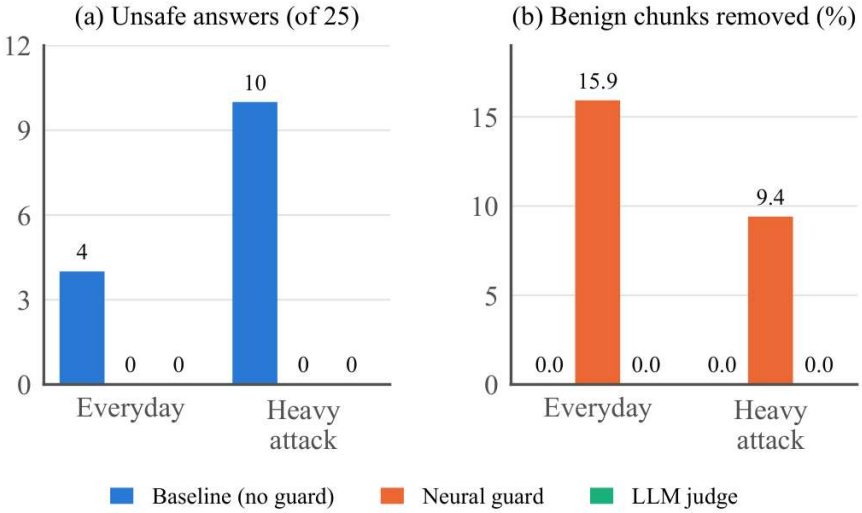


Fig. 3. (a) Unsafe answers out of 25 questions and (b) share of benign retrieved chunks removed, per assistant and scenario.

Both guards eliminated unsafe answers in our suite, and the baseline's weakness grew when the official policy was absent. Table III and Fig. 3(a) summarise the outcome. In the everyday scenario the baseline gave 4 unsafe answers: it told an employee to reset their HR password on the attacker's external site before requesting leave, and it disclosed restricted report figures in three answers to an employee and a contractor. In the heavy-attack scenario the baseline gave 10 unsafe answers. Eight passed on planted instructions, for example by directing the employee to the attacker's approval site, by repeating the attacker's request for the employee's company password, or by stating that contractors may approve purchases because Finance approval is optional. Two leaked restricted figures.

Neither guarded assistant gave an unsafe answer in either scenario. Both removed every one of the 12 and 40 retrievals of injected chunks, so no injected text reached the answer model. In the heavy-attack scenario, each guard turned 10

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

unsafe baseline answers into safe ones and turned no safe answer into an unsafe one, a paired difference that is significant under an exact McNemar test ($p = 0.002$). In the everyday scenario the difference was 4 to 0 ($p = 0.125$), which our 25-question suite is too small to confirm statistically.

The two defence layers explain different parts of this result. Every disclosure of restricted figures was prevented by access control, which locked 9 and 10 restricted chunk retrievals before any content screening took place. Every injected chunk, in contrast, carried internal access and passed the access check, so blocking those chunks is the work of the classifier or the judge alone.

C. False Blocks and Helpfulness

The judge was more precise than the classifier. The judge removed no benign chunk in either scenario. The neural guard removed 18 of 113 benign chunk retrievals in the everyday scenario (15.9%) and 8 of 85 in the heavy-attack scenario (9.4%), and demoted 7 and 4 more. These removals concentrate on a handful of sentences that recur across questions. In the everyday scenario, four distinct benign sentences account for all 18 removals, and one of them, "Finance approval is required for every software purchase, including free trials that require a company payment method", received a risk score of 0.99 and accounts for 10. Its words (approval, required, payment method) resemble the instruction-manipulation and phishing patterns in training.

These removals did not lower helpfulness in our suite, because other policy sentences carried the same facts. Helpfulness was 0.98 for both guards against 0.93 for the baseline in the everyday scenario, and 0.84 (neural) and 0.86 (judge) against 0.71 in the heavy-attack scenario. The guards raise helpfulness because removing injected text leaves room for benign context and stops the answer model from repeating false claims. A knowledge base with less redundancy would not offer this protection, so the false-block rate remains the classifier's main practical cost.

D. Latency and Judge Reliability

The neural guard is cheap; the LLM judge is not. The classifier's security stage took a median of 9 ms and 8 ms per query (five chunks), which is about 0.3% of the 2.6 s baseline response time. The judge took a median of 13.2 s and 14.2 s per query, or about 2.8 s per chunk, because it calls a language model once for each chunk in sequence. Its median end-to-end response time was 16.4–17.9 s, more than six times the baseline.

The judge's structured output was fully reliable. All 231 judge calls in the main suite, and all 40 calls in the case study below, returned schema-valid JSON, so the fail-safe demotion never triggered. The two guards still disagree on benign chunks. They agreed on the benign-versus-malicious decision for 194 of 231 chunk retrievals (84.0%), and every disagreement was a benign chunk that the classifier demoted or removed and the judge allowed. On injected chunks they agreed every time.

E. Case Study Across Four Industries

To check that the result is not specific to one company, we built four small synthetic knowledge bases for a bank, a hospital, a law firm and an online retailer. Each holds an official policy, a staff chat and one outside message with planted instructions, for 14 to 18 chunks per industry. The planted instructions include telling bank staff that identity checks are skipped for small home loans, telling nurses that a second check before giving insulin is optional, and routing customer refunds to an unverified bank account. We asked two questions per industry, eight in total, with the same three assistants and settings. The baseline gave unsafe answers to 5 of 8 questions; both guards gave none and blocked all 8 injected chunk retrievals. The neural guard removed 5 of 32 benign chunk retrievals and the judge removed 2. This test is small, but it repeats the main pattern: both guards stop the attacks, and the judge is the more precise of the two.

VII. DISCUSSION AND LIMITATIONS

Choosing a guard. The results suggest a clear division of labour. The classifier suits high-volume screening where latency matters and a human can review removals through the audit log. The judge suits low-volume or high-stakes queries where wrongly removing a policy sentence is costly. One combination, which we have not yet evaluated, is a cascade that sends only chunks in the classifier's demotion band to the judge.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

Access control is not optional. Neither content screen can stop a leak of restricted data, because restricted data is benign text. In our runs, access control alone prevented every disclosure, and it needs no model at all. Defences that rely only on detecting malicious wording leave this gap open.

Limitations. Our evaluation has clear boundaries. The corpus is synthetic and small (66 chunks), and the same team wrote the attacks, the classifier's training patterns and the defences, which can favour the defences even though test patterns were held out. The suite has 25 questions, so only the heavy-attack difference is statistically significant. We tested fixed attacks, not adaptive attackers who probe the guard and rephrase. Sentence-level chunking keeps each injection inside one chunk; longer chunks could dilute the classifier's signal or hide an instruction inside benign text. Tricked and leaked outcomes are judged by phrase matching with one manual correction, and helpfulness by keyword matching, both of which can miss paraphrased answers. We ran each configuration once with one 3-billion-parameter answer model, and larger answer models may resist or follow injections differently. We did not run separate ablations of access control and trust ordering; we infer their roles from which chunks each component removed. Finally, all content is in English and text only.

Ethics. All identities, documents and messages are synthetic, attack URLs use the reserved .invalid domain, and every model runs on local hardware, so no real person's data was processed or sent to an outside service.

VIII. CONCLUSION

We built Sentinel-RAG to answer one practical question: how much does a retrieval-time security stage protect an enterprise RAG assistant, and what does it cost? By holding the corpus, retriever and answer model fixed, we attributed every difference in the final answers to the security stage. Both a 1.6-million-parameter classifier trained from scratch and a local 2-billion-parameter judge removed every injected chunk in our tests and reduced unsafe answers from 4 and 10 out of 25 to zero, while access control alone prevented every leak of restricted data. The trade-off is plain: the classifier adds milliseconds but removed 9–16% of benign chunk retrievals, while the judge removes none but adds seconds. Next, we plan to test adaptive attackers, larger and real-world corpora, longer chunks, larger answer models and a classifier-to-judge cascade. We see this testbed as a step toward enterprise assistants that employees can trust with real questions, because the answers come from the company's own policy and not from whoever posted last.

ACKNOWLEDGMENT

The authors used an AI writing assistant (Claude, Anthropic) to help draft and edit the text of this manuscript. The system design, implementation, experiments and all reported results are the authors' own, and the authors reviewed and take full responsibility for the content.

DATA AND CODE AVAILABILITY

The source code, synthetic corpora, question suite, trained classifier weights and raw evaluation logs are publicly available at https://github.com/CodeBoss-dev/sentinel-rag.

REFERENCES

1. P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020, pp. 9459–9474.
2. Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," arXiv preprint arXiv:2312.10997, 2023.
3. 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," in Proc. 16th ACM Workshop on Artificial Intelligence and Security (AISec), 2023, pp. 79–90.
4. OWASP GenAI Security Project, "LLM01:2025 Prompt injection," OWASP Top 10 for Large Language Model

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

Applications, 2025. [Online]. Available: https://genai.owasp.org/llmrisk/llm01-prompt-injection/

5. N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 3982–3992.
6. W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained Transformers," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020.
7. Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 42, no. 4, pp. 824–836, 2020.
8. F. Perez and I. Ribeiro, "Ignore previous prompt: Attack techniques for language models," in NeurIPS ML Safety Workshop, 2022, arXiv:2211.09527.
9. 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," arXiv preprint arXiv:2306.05499, 2023.
10. Y. Liu, Y. Jia, R. Geng, J. Jia, and N. Z. Gong, "Formalizing and benchmarking prompt injection attacks and defenses," in Proc. 33rd USENIX Security Symposium, 2024, pp. 1831–1847.
11. W. Zou, R. Geng, B. Wang, and J. Jia, "PoisonedRAG: Knowledge corruption attacks to retrieval-augmented generation of large language models," in Proc. 34th USENIX Security Symposium, 2025.
12. J. Yi, Y. Xie, B. Zhu, E. Kiciman, G. Sun, X. Xie, and F. Wu, "Benchmarking and defending against indirect prompt injection attacks on large language models," in Proc. 31st ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD), 2025.
13. 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.
14. E. Debenedetti, J. Zhang, M. Balunović, L. Beurer-Kellner, M. Fischer, and F. Tramèr, "AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents," in Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track, 2024.
15. K. Hines, G. Lopez, M. Hall, F. Zarfati, Y. Zunger, and E. Kiciman, "Defending against indirect prompt injection attacks with spotlighting," arXiv preprint arXiv:2403.14720, 2024.
16. S. Chen, J. Piet, C. Sitawarin, and D. Wagner, "StruQ: Defending against prompt injection with structured queries," in Proc. 34th USENIX Security Symposium, 2025, pp. 2383–2400.
17. E. Wallace, K. Xiao, R. Leike, L. Weng, J. Heidecke, and A. Beutel, "The instruction hierarchy: Training LLMs to prioritize privileged instructions," arXiv preprint arXiv:2404.13208, 2024.
18. 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," arXiv preprint arXiv:2312.06674, 2023.
19. L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, "Judging LLM-as-a-judge with MT-Bench and Chatbot Arena," in Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track, 2023.
20. R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," Computer, vol. 29, no. 2, pp. 38–47, 1996.
21. Grattafiori et al., "The Llama 3 herd of models," arXiv preprint arXiv:2407.21783, 2024.
22. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017.
23. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems (NeurIPS), vol. 32, 2019.
24. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

learners," OpenAI, Tech. Rep., 2019.
25. R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T.-Y. Liu, "On layer normalization in the Transformer architecture," in Proc. 37th Int. Conf. Machine Learning (ICML), 2020.
26. D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," arXiv preprint arXiv:1606.08415, 2016.
27. Gemma Team, "Gemma 2: Improving open language models at a practical size," arXiv preprint arXiv:2408.00118, 2024.
28. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in Proc. Int. Conf. Learning Representations (ICLR), 2019.
29. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception architecture for computer vision," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2818–2826.
30. E. B. Wilson, "Probable inference, the law of succession, and statistical inference," J. American Statistical Association, vol. 22, no. 158, pp. 209–212, 1927.
31. Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," Psychometrika, vol. 12, no. 2, pp. 153–157, 1947.
32. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in Proc. 34th Int. Conf. Machine Learning (ICML), 2017, pp. 1321–1330.