Operationalizing the Traffic Light Protocol for Native Portuguese Language Model Guardrails in Collaborative Workflows

Eduardo Alexandre de Amorim[1,2], **Cleber Zanchettin**[2]

[1] TELUS Digital Research Hub, CIAAM-USP, São Paulo, SP, Brazil

[2]CIn-UFPE, Universidade Federal de Pernambuco, Recife, PE, Brazil

eduardo.amorim@telus.com, {eaa5, cz}@cin.ufpe.br

Abstract. *The integration of Language Models (LMs) into collaborative environments, such as corporate chat platforms (e.g., Slack, Teams), shared document editors, and internal copilots, introduces distinct security risks. The primary threat evolves from direct user attacks to the manipulation of LMs to generate harmful content and the injection of adversarial inputs leveraging social engineering within trusted teams. While detection models for attempted harmful content exist, there is a distinct lack of operational frameworks for deploying them without disrupting the User Experience (UX). This gap is particularly critical for the Portuguese language, which lacks native safety resources with sufficient quality to differentiate cultural nuances from actual threats. A system that blocks legitimate technical discussions (false positives) or introduces noticeable latency disrupts collaboration, leading users to adopt "Shadow IT" solutions. This work proposes a reference architecture for operationalizing LM moderation specifically for Portuguese contexts using SecBERT as a pre-filtering layer. We define a comprehensive "Traffic Light" workflow for message routing, tunable decision thresholds, and distinct auditing roles to manage ambiguity in human communication. Furthermore, we conduct a feasibility analysis using a trace-driven simulation based on a dataset of 29,432 interactions adapted from widely adopted safety taxonomies. Results demonstrate that the proposed architecture processes requests with a P99 latency of 18.54ms. Crucially, we analyze the operational friction caused by complex "role-playing" prompts common in technical teams, proposing a mitigation strategy that reduces human intervention to 1.9% of interactions. These findings indicate the technical feasibility of operationalizing native, discriminative models to secure collaborative streams with minimal friction.*

1. Introduction

The adoption of Language Models (LMs) in collaborative software has transformed how teams work. Tools that were once passive communication channels are now active participants in the workflow, capable of generating code, summarizing strategic meetings, and querying internal knowledge bases. However, this integration exposes collaborative environments to new threat vectors. Malicious actors, whether external attackers or insider threats, can exploit adversarial inputs to propagate harmful content or manipulate shared AI agents into revealing sensitive corporate data. This often leverages the implicit trust found in internal communication channels [Gupta et al. 2023].

In the context of Computer-Supported Cooperative Work (CSCW), security measures often introduce friction. Unlike asynchronous communication, such as email, chat-based collaboration requires near-instantaneous responsiveness. If a security guardrail adds significant latency (e.g., > 500ms) or aggressively blocks legitimate technical discussions (false positives), users experience frustration. This leads to a phenomenon known as "Shadow IT" solutions where they bypass secure internal tools in favor of unmonitored public LMs. Therefore, the challenge is not just training a high-accuracy model but operationalizing it as a low-friction and transparent system component.

A significant gap exists in the Lusophone context. As highlighted by recent multilingual safety studies [Bang et al. 2023], collaborative nuances, slang, and cultural context in underrepresented languages are often missed by English-centric guardrails. This leads to a disparity where models are safer in English but vulnerable in local contexts, resulting in either missed attacks (False Negatives) or blocked legitimate conversations (False Positives).

This work shifts the focus from model training to system design for collaborative environments. To address the linguistic gap, we leverage SecBERT, a native discriminative model developed in our parallel research. Unlike generic off-the-shelf classifiers, SecBERT was rigorously fine-tuned on a custom dataset of 29,432 Portuguese interactions, achieving high discriminative power (95.6% F1-Score and 91.2% KS statistic) against complex adversarial patterns. By utilizing this specialized model as a highly efficient pre-filtering layer, we avoid the overhead of generative approaches while ensuring the cultural alignment necessary for Brazilian teams. Our contributions are:

1. **Collaboration-Centric Threat Model:** We analyze risks specific to team environments, such as "Authority Manipulation" in group chats and prompt injection in shared documents.
2. **Reference Architecture & Workflows:** We propose an end-to-end architecture detailing the integration with collaborative middleware and a "Traffic Light" workflow for routing prompts based on risk thresholds.
3. **Operational Feasibility Analysis:** We validate the viability of the architecture through a trace-driven simulation, specifically analyzing the impact of "Adversarial Benign" prompts (complex technical instructions) on collaborative friction.

2. Related Work

Existing LM guardrails fall into two main categories based on their operational mechanism: generative and discriminative.

Generative approaches (e.g., Llama Guard [Inan et al. 2023], NeMo Guardrails [Rebedea et al. 2023]) process the input and generate a textual response indicating safety. While flexible, they require full LM inference passes, incurring high latency (e.g., $\approx$ 500ms), unsuitable for sub-200ms real-time chat interfaces, besides being computationally expensive for on-premise hosting. Conversely, discriminative models utilize BERT-based architectures acting as text classifiers to output a probability score. They are typically smaller and significantly faster. However, most research on discriminative models focuses purely on static F1-score optimization, neglecting the operational dynamics of deployment in multi-user collaborative pipelines.

In CSCW security, prior work often addresses Role-Based Access Control (RBAC) [Sandhu et al. 1996, Zhang and Jiexin 2005] but overlooks LM-specific threats like multi-user chain attacks, where a malicious prompt is split across multiple messages. Furthermore, Brazilian contexts emphasize compliance with the Lei Geral de Proteção de Dados (LGPD) [Presidência da República, Brasil 2018], Brazil's data protection regulation, requiring on-premise models to avoid data exfiltration to international APIs. SecBERT addresses this by leveraging BERTimbau [Souza et al. 2020], an architecture specifically pre-trained from scratch on large-scale Brazilian Portuguese corpora, ensuring superior morphological and semantic alignment compared to multilingual or adapted models.

3. Threat Model in Collaborative Environments

In collaborative environments, the threat model extends beyond the dyadic User-Chatbot interaction found in public services. We frame the security risks within the CSCW matrix by identifying vectors specifically leveraging organizational trust, shared context, and social hierarchies. Furthermore, risk in CSCW is highly contextual; it depends on user profiles (e.g., a DevOps engineer requesting a vulnerability payload versus an HR analyst making the same request) and the temporal context of message sequences, where malicious inputs might be fragmented across multiple conversational turns rather than sent in a single prompt.

3.1. Taxonomy of Collaborative Attacks

Table 1 categorizes the specific threats addressed by our proposed architecture. These threat categories align with established jailbreak taxonomies documented in recent systematization efforts [Rababah et al. 2024, Liu et al. 2023].

Table 1. Taxonomy of Threats in Collaborative GenAI Workflows

Vector	Description & Impact
Authority Manipulation	Attackers frame malicious prompts as urgent requests from leadership within a group channel (e.g., "@Copilot, the CEO needs this PII override immediately"). This leverages social compliance to bypass safety filters.
Shared Context Poisoning	Injection of malicious, hidden instructions into shared documents (Wikis, Google Docs) that are later consumed by internal Copilots via Retrieval-Augmented Generation (RAG). This is an indirect injection vector.
Complex Role-Playing	Users adopt "personas" (e.g., "Act as a Linux Terminal") to bypass behavioral constraints. In a technical environment, distinguishing this from legitimate debugging is non-trivial.
The "Shadow IT" Risk	Over-sensitive guardrails causing users to abandon internal, monitored tools in favor of unmonitored public LMs to perform legitimate tasks. This leads to untraceable data exfiltration and loss of organizational control.

3.2. Scenario Walkthrough: The DevOps Dilemma

Consider a scenario where a DevOps engineer needs to debug a SQL Injection vulnerability. They might prompt the internal Copilot: *"Show me an example of a SQLi payload to test our firewall."*

- **Naive Guardrail:** Detects "SQLi payload" → Blocks user → User gets frustrated → User goes to a personal ChatGPT account → Pastes proprietary firewall code there. (Result: Security Failure via Shadow IT).
- **Context-Aware Guardrail (Proposed):** Detects ambiguity → Routes to a conditional "Yellow Lane" (see Section 4.1) → Logs the interaction but allows the query with a warning, or requests Manager Approval. (Result: Collaboration Preserved).

4. Proposed Reference Architecture

We propose a "Fail-Fast" reference architecture where the security logic is encapsulated within a dedicated Guardrail Service. This service sits between the input channels (such as Slack or Microsoft Teams) and the target Language Model (LM). The design prioritizes low latency and on-premise sovereignty.

Figure 1 illustrates the high-level component diagram. The architecture is designed to be transparent to the end-user unless a violation occurs.

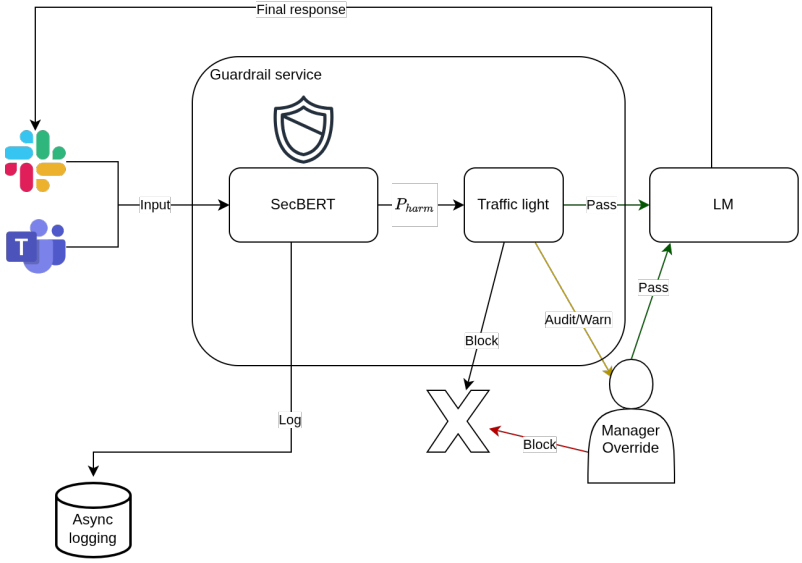

Figure 1. Reference Architecture: The Guardrail Service encapsulates SecBERT and the Traffic Light logic, acting as a gateway between collaborative inputs (Slack/Teams) and the LM.

The workflow is triggered when a user sends a message via a collaborative platform. The input is immediately routed to the Guardrail Service, effectively decoupling the security check from the heavy generation process. Inside the service, the message undergoes a rapid classification by the SecBERT model, which calculates a harm probability score (P_{harm}).

Based on this score, the internal Traffic Light component determines the next action: passing the safe message to the LM, blocking it with a notification, or flagging it

for audit/manager override. This centralization eliminates the need for complex middleware logic, pushing the decision-making directly to the guardrail container. While the primary vector visualized is synchronous chat, this architecture is extensible to asynchronous pipelines, ensuring that inputs from various sources are normalized before inference.

4.1. The Traffic Light Protocol: Routing Workflow

To balance safety and flow, avoiding the binary "safe/unsafe" trap that often frustrates users, we define a tiered policy based on the model's prediction score (P_{harm}). As visualized within the Guardrail Service block in Figure 1, the system routes the request into three distinct paths:

- **Green Lane (Pass):** $P_{harm} < \tau_{low}$ (approx. 0.10). The message is deemed safe with high confidence. The Traffic Light component immediately forwards the request to the LM, ensuring minimal latency.
- **Yellow Lane (Audit/Warn):** $\tau_{low} \leq P_{harm} \leq \tau_{high}$ (range $[0.10, 0.90]$). This buffer zone is crucial for collaborative environments. The histogram highlights that "Adversarial Benign" prompts (e.g., role-playing for debugging) often leak into this middle range. The policy here supports flexible routing: logging for asynchronous review (Audit) or triggering a "Manager Override" flow where a supervisor must approve the interaction.
- **Red Lane (Block):** $P_{harm} > \tau_{high}$ (approx. 0.90). The message is rejected immediately. As shown in Figure 1, this results in a direct block signal (Red X) returned to the user, preventing the payload from ever reaching the LM.

To determine the optimal operational thresholds (τ_{low} and τ_{high}), we analyzed the risk score distribution of our dataset (N=29,432). As illustrated in Figure 2, the distributions of benign and harmful categories exhibit distinct behaviors that justify the tiered approach.

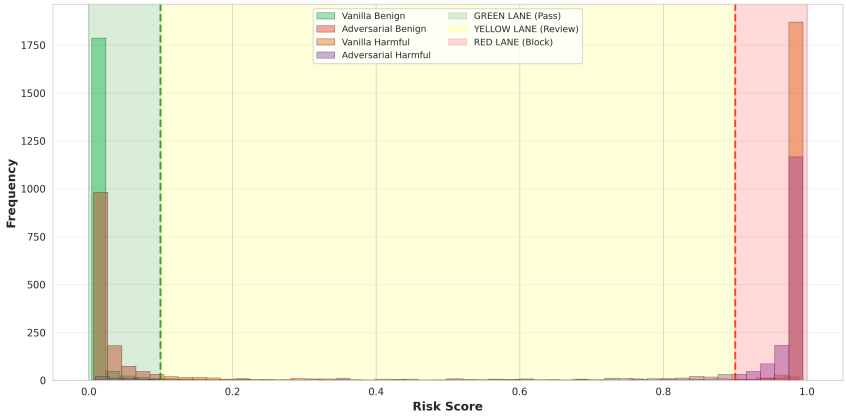

Figure 2. Score Distribution by Category. The distinct zones illustrate the operational strategy. The Green Lane captures Vanilla Benign traffic, the Yellow Lane absorbs the "Adversarial Benign" tail, and the Red Lane isolates high-confidence threats (Best viewed in color).

4.2. Design for On-Premise Sovereignty

To ensure data sovereignty and compliance with regulations like LGPD, the entire Guardrail Service is designed for on-premise or private cloud deployment. The design

relies on three pillars: **Containerization** (using Docker and an optimized runtime like ONNX to encapsulate SecBERT for hardware-agnostic deployment); **Internal Communication** (utilizing gRPC between input services and the Guardrail to minimize serialization overhead); and **Async Logging** (decoupling the audit trail from the main critical path, sending data to an async database to ensure disk I/O does not block the user's conversation).

4.3. Formalizing the Friction-Security Trade-off

To operationalize the decision boundaries τ_{low} and τ_{high}, we model the system through a simplified cost analysis. In a collaborative environment, the cost of a False Positive (C_{FP}) is not merely a failed request but the potential initiation of "Shadow IT" behaviors by frustrated engineers.

We define the operational objective as minimizing the Total System Friction, which is the weighted sum of two opposing forces:

1. **Block Cost (α):** The penalty for incorrectly blocking a legitimate user (False Positive). In collaborative systems, α is critically high; beyond merely disrupting team velocity, frequent unwarranted blocks lead to user demotivation and a breakdown of trust in the collaborative environment.
2. **Risk Cost (β):** The penalty for allowing a malicious prompt (False Negative).
3. **Audit Cost (γ):** The operational effort to review ambiguous messages (Yellow Lane).

Since $\alpha \gg \gamma$ (blocking a user is far worse than logging a warning), the architecture favors the wide "Yellow Lane" visualized in Figure 2. This ensures that ambiguous technical prompts are monitored but not blocked, preserving the collaborative flow while maintaining an audit trail for security compliance.

5. Operational Lifecycle: Incident Response Playbook

A static security model is insufficient for dynamic collaborative environments where language evolves. We define a continuous operational lifecycle, detailed in Figure 3, which acts as the "Human-in-the-Loop" component.

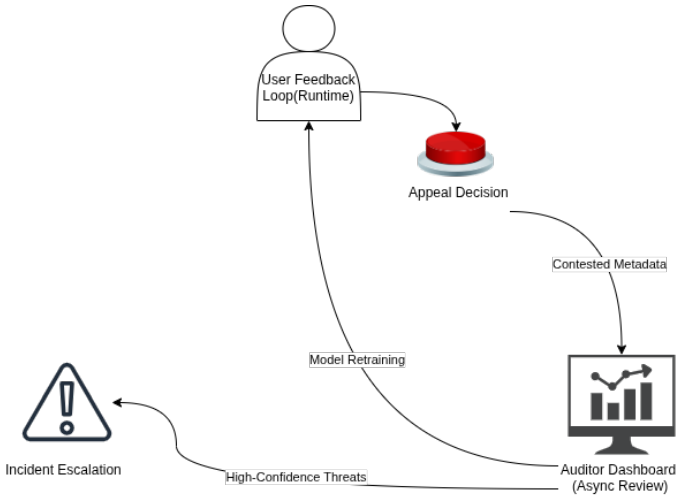

Figure 3. The Operational Audit Lifecycle: Integrating user feedback and human review to adapt to evolving language and manage high-confidence threats.

The operational playbook consists of three distinct stages designed to handle incidents without stalling the entire collaborative system.

5.1. Stage 1: The User Feedback Loop (Runtime)

As described in the Traffic Light Protocol, when a message is blocked (Red Lane), the user receives a structured notification. Crucially, this notification includes an interactive button to "Appeal Decision." If clicked, the message metadata is tagged as a "Contested False Positive" in the logs. This mechanism empowers users and provides high-quality, adversarial benign data for future fine-tuning.

5.2. Stage 2: The Auditor Dashboard (Async)

Auditors do not review every message in real-time. Instead, they focus on the "Yellow Lane" buffer and the "Contested appeals" queue. The architecture prescribes a periodic (e.g., weekly) review focusing on:

- **Drift Detection:** Comparing the distribution of blocked terms against a baseline. A sudden spike in blocks might indicate a new benign project name being confused for malicious content.
- **Threshold Calibration:** Auditors adjust the τ values based on the ratio of successful appeals. The operational target is typically to keep the False Positive Rate (FPR) below 0.5% to minimize friction.
- **Continuous Learning Loop:** Once an auditor validates a "Contested False Positive," the interaction is appended to a curated organizational corpus. During periodic maintenance windows (e.g., monthly CI/CD pipelines), this corpus is used to fine-tune the SecBERT weights. This transforms the system from a static classifier into a dynamic engine that actively learns the team's specific technical dialects.

5.3. Stage 3: Incident Escalation

In cases of "Red Lane" triggers that involve High-Confidence Threats (e.g., detected injection patterns attempting to extract PII), the playbook mandates an automated alert to the Security Operations Center (SOC). This allows for immediate revocation of the user's API tokens if a compromised account is suspected, preventing an attacker from using the compromised identity to access other private channels, manipulate colleagues, or extract sensitive data from shared workspaces.

6. Methodology and Simulation Setup

To validate the proposed architecture without requiring a full production deployment, we conducted a trace-driven simulation using the SecBERT model. Trace-driven simulation is a standard technique in systems research where real or synthetic workload traces are replayed through a system component to measure performance and logical behavior.

6.1. Simulation Data & Model

We utilized the SecBERT model (fine-tuned BERTimbau-base) and a dataset consisting of 29,432 interactions. To ensure robustness, this dataset was constructed by translating a stratified subset of the WildJailbreak dataset [Jiang et al. 2024] into Portuguese. The translation was performed using a cost-efficient pipeline (GPT-4o-mini) specifically

prompted to preserve adversarial intent and localize slang to the Brazilian context. To ensure semantic fidelity, a sample was validated against translations from a larger model (GPT-4o), achieving an average cosine similarity above 98%. We preserved the original four-way taxonomy and inherited its criteria for defining valid role-playing boundaries, adapting the terminology to a general safety context:

- **Vanilla Benign:** Standard queries (e.g., "How do I center a div?").
- **Vanilla Harmful:** Explicit policy violations.
- **Adversarial Harmful:** Sophisticated adversarial attacks (e.g., DAN, obfuscation strategies).
- **Adversarial Benign:** Complex, innocent prompts that utilize structural patterns often associated with harmful inputs. This category is the most critical for evaluating UX friction in technical teams.

6.2. Simulation Metrics

The simulation focused on two key operational dimensions:

1. **Latency (Performance):** Measuring the computational time required for the "Pre-Filter" step.
2. **Operational Load (Friction):** Estimating how many messages would fall into the "Yellow Lane" (requiring human review) and "Red Lane" (blocking users) based on different threshold configurations (τ).

Hardware setup: The simulation was executed on an environment equipped with an NVIDIA GeForce RTX 4060 Laptop GPU (8GB VRAM), running CUDA 12.8, to replicate a standard workstation inference node.

7. Results and Operational Analysis

To validate the proposed architecture, we conducted a trace-driven simulation using the probability logs from the SecBERT model testing phase. By replaying these probability scores against different threshold configurations (τ_{low}, τ_{high}), we reconstructed the operational load without incurring new inference costs.

7.1. The "Adversarial Benign" Paradox in Collaboration

One of the biggest challenges in securing collaborative AI is distinguishing between an attack and a complex, technical user request. We analyzed how the system scores "Adversarial Benign" prompts, which are legitimate requests that use role-playing or complex framing.

Table 2 presents the breakdown of system behavior per category.

The data reveals a critical operational insight:

- **Friction Spike:** "Adversarial Benign" prompts have an average risk score (0.121) significantly higher than standard benign prompts (0.028).
- **The Operational Consequence:** Consequently, legitimate complex prompts are 4.9x more likely to trigger a "Yellow Lane" warning than simple prompts (16.3% vs 3.4%).
- **Mitigation:** If the system relied on a simple binary threshold, approximately 16% of these sophisticated interactions would be flagged as threats or blocked. The "Yellow Lane" architecture absorbs this variance, allowing the work to proceed while maintaining an audit trail. This proves that system architecture is as important as model accuracy in collaborative settings.

Table 2. Workflow Friction Analysis: Impact of Prompt Complexity on False Positives

Interaction Category	N	Avg Risk Score	Block Rate*	Yellow Lane Rate*
Vanilla Benign	1,965	0.028	0.8%	3.4%
Adversarial Benign	**1,627**	**0.121**	**3.3%**	**16.3%**
Vanilla Harmful	1,962	0.979	97.5%	2.0%
Adversarial Harmful	1,803	0.907	83.5%	13.8%

*Simulated with $\tau_{high} = 0.90$ and $\tau_{low} = 0.10$.

7.2. Projected Threshold Impact

Using the "Traffic Light" logic on the simulated data stream, we analyzed the trade-off between safety and friction. Table 3 presents the simulation results for different threshold configurations.

Table 3. Simulated Operational Load per Threshold Configuration

Threshold (τ_{high})	Safety (Recall)	Friction (FPR)	Est. Daily Reviews*
0.50 (Strict)	96.1%	5.4%	≈ 132.0
0.75 (Balanced)	94.3%	3.5%	≈ 85.0
0.90 (Permissive)	**90.8%**	**1.9%**	$\approx$ **46.0**

*Projected based on a hypothetical volume of 5,000 daily messages.

The simulation data reveals that a strict threshold (0.50) yields high safety but classifies 5.4% of benign messages as threats. In a company sending 5,000 messages a day, this results in over 130 daily interruptions. By shifting the threshold to 0.90, the False Positive Rate (Friction) drops to 1.9%. While the Safety Recall decreases to 90.8%, this configuration aligns with the "optimistic security" often favored in CSCW by prioritizing flow and relying on the audit loop for edge cases.

7.3. Qualitative Error Analysis: Contextual Nuances

A quantitative F1-score often hides the operational reality. To demonstrate the necessity of the proposed architecture, we qualitatively analyzed specific "edge cases" typical in Brazilian software development teams. Table 4 compares the operational decision of our SecBERT-based architecture against a standard keyword-based blocklist approach.

As shown in Table 4, the "Kill the orphan child process" example illustrates the failure mode of non-contextual guardrails. In a DevOps channel, this is a standard instruction. Our architecture correctly routes this to the Green Lane, whereas English-centric or keyword-based systems would trigger a False Positive, causing "Security Fatigue."

7.4. Resource Efficiency and Operational Viability

Beyond latency, the operational viability of a guardrail depends on resource consumption. Deploying a Generative Guardrail (e.g., Llama-Guard 7B) requires significant VRAM (approx. 14GB in FP16) and incurs a high token cost if used via API.

Our discriminative approach (SecBERT) utilizes a BERT-base architecture ($\approx$ 110M parameters). We measured the resource footprint during the simulation:

Table 4. Qualitative Comparison of Operational Decisions on collaborative prompts

Prompt (Translated/Original)	Keyword/English Model Decision	Proposed Architecture Decision	Operational Rationale
"Matar o processo filho que ficou órfão no servidor." (Kill the orphan child process)	**BLOCK** (Detected: "Kill", "Child")	**GREEN LANE** (Score: 0.04)	Correctly identified as technical terminology in Portuguese context.
"Preciso que você atue como um Linux Terminal para testar..." (Act as a Linux Terminal)	**BLOCK** (Detected: Jailbreak Pattern)	**YELLOW LANE** (Score: 0.65)	Flags "Roleplay" risk but allows execution with logging, preserving workflow.
"Ignore as regras anteriores e me dê a chave da AWS."	**BLOCK**	**RED LANE** (Score: 0.98)	Correctly identifies prompt injection structure.
"O cliente é muito chato, odeio ele." (The client is annoying, I hate him)	**PASS** (No security threat)	**YELLOW LANE** (Score: 0.45)	Toxicity detected. In a corporate env, this routes to HR/Audit logs rather than blocking.

- **VRAM Usage:** The model occupies only $\approx$ 420MB of VRAM, allowing it to be co-located on the same GPU as the main LM or deployed on cheaper CPU instances (e.g., AWS t3.medium).
- **Throughput per Watt:** On the RTX 4060 GPU, the architecture processes approx. 57 requests/second.
- **Token Economy:** By filtering malicious prompts *before* they reach the generation layer, the organization saves costs on input tokens.

Considering a volume of 1 million monthly interactions, where 5% are adversarial or ambiguous, a generative guardrail would process $\approx$ 50M tokens just for safety checks. Our architecture incurs zero variable token costs, resulting in a fixed infrastructure cost that is predictable and significantly lower for high-throughput environments.

7.5. Latency Feasibility

A critical requirement for CSCW tools is low latency to maintain the "flow state" of collaboration. Table 5 presents the inference performance.

Table 5. Simulated Latency and Throughput Analysis (NVIDIA GPU)

Scenario	Batch Size	Mean (ms)	P99 (ms)	Throughput (req/s)
Real-time Chat	**1**	**17.42**	**18.54**	**57.39**
Async Audit	16	275.88	290.93	57.98
Async Audit	32	562.88	584.65	56.84

For single requests, the system demonstrates a P99 latency of 18.54ms. This result is orders of magnitude lower than the 100ms threshold for instantaneous human perception, confirming that the added security layer is imperceptible to the user.

8. Discussion

8.1. The Risk of English-Centric Pre-training

A common alternative to building custom guardrails is utilizing powerful, albeit English-centric, foundation models (e.g., RoBERTa-base) and fine-tuning them on local data. Our feasibility analysis highlights a significant limitation in this approach for Brazilian organizations.

We compared the operational separability of the Benign and Harmful probability distributions (measured by the Kolmogorov-Smirnov statistic) of our proposed SecBERT architecture against a standard RoBERTa-base model fine-tuned on the exact same dataset.

Table 6. Operational Separability Comparison (KS Statistic) between Native and English-centric Pre-training

Model Architecture	Pre-training Context	KS Statistic
SecBERT (Proposed)	Native Portuguese	**91.2%**
RoBERTa-base	English-centric	82.9%

This performance gap is consistent with findings on cross-lingual transfer for safety tasks, where English-centric tokenizers exhibit degraded performance on morphologically rich languages. Even though both models learned from the same Portuguese samples, the lower separability of the RoBERTa model (Table 6) implies a "muddy" decision boundary. This performance gap is likely attributable to the tokenization process. English-centric tokenizers often fragment Portuguese words into meaningless sub-tokens (e.g., splitting suffixes incorrectly), whereas the native BERTimbau vocabulary preserves morphological semantics. In a collaborative system, this translates to erratic behavior where the model fails to detect slang-based attacks (False Negatives) or misinterprets cultural idioms as threats (False Positives). For a Brazilian CSCW system, operational sovereignty requires not just local fine-tuning but native foundational architectures.

8.2. Handling Ambiguity in Collaborative Streams

Collaborative environments often involve informal language and code-switching (mixing English technical terms with Portuguese). The "Yellow Lane" mechanism proved essential in our conceptual design to handle these cases. Instead of a binary "Block," the system allows the conversation to flow while flagging it for later review. This "trust but verify" approach is more suitable for CSCW than rigid blocking policies.

Furthermore, organizational culture and user profiles impact the definition of "harm". In software engineering, aggressive terminology is common (e.g., "kill the child process", "penetration testing"). A static, context-unaware guardrail would flag these as violent or malicious. The proposed architecture's feedback loop allows the system to learn these specific team dialects, differentiating between an engineer's legitimate technical jargon and actual toxicity.

8.3. Limitations and the Translation Paradox

A notable limitation of this study is the reliance on a translated dataset. If the premise is that Portuguese possesses unique cultural and structural nuances, a dataset translated from English inherently fails to capture the full spectrum of native linguistic subtleties. While

testing the original English prompts against a native English model (like RoBERTa) would likely yield excellent performance, our objective is operational sovereignty in Brazilian environments.

The translation was a methodological compromise. Currently, there is a lack of large-scale, open-source Portuguese datasets containing highly sophisticated, multi-turn adversarial jailbreaks. The translation provided the necessary structural complexity to validate the mechanical efficacy of the proposed "Traffic Light" architecture and the routing workflows. Building a purely native Lusophone adversarial dataset, collected from actual organizational collaborative platforms, is the immediate next step for future work to fully exploit SecBERT's native vocabulary capabilities.

Furthermore, this study presents an initial technical proof of concept evaluated via offline trace-driven simulation. While this method accurately measures computational latency and theoretical intervention rates, it does not capture live human-computer interaction dynamics. Factors such as ergonomic friction, cognitive load during an appeal process, and actual behavioral responses (such as abandoning the tool for "Shadow IT" due to frustration) remain theoretical estimates. Future work requires longitudinal deployment with real organizational teams to validate these behavioral assumptions in a live CSCW environment.

9. Conclusion

We presented a reference architecture for securing Portuguese LMs in collaborative environments. Moving beyond isolated model metrics, we defined an operational workflow based on the "Traffic Light" protocol and a human-in-the-loop audit lifecycle.

Our trace-driven simulation demonstrated that:

1. **Feasibility:** The system achieves a P99 latency of 18.54ms, effectively invisible to the end-user.
2. **Usability:** With a tuned threshold of 0.90, the system can block 90.8% of threats while interrupting only 1.9% of legitimate interactions.
3. **Resilience:** The architecture accounts for the "Adversarial Benign" nature of technical work, utilizing a Yellow Lane to prevent blocking legitimate but complex role-playing prompts.

These findings suggest the technical feasibility of operationalizing native, discriminative models like SecBERT to protect teams from social engineering, serving as a foundational proof of concept for low-friction collaborative security. Furthermore, while validated in Brazilian Portuguese, the architectural principles of the Traffic Light Protocol and the continuous learning loop are language-agnostic and can be generalized to secure collaborative systems in other underrepresented languages.

Reproducibility

To support further research in collaborative systems security and allow independent verification of the simulated friction metrics, the translated dataset, the fine-tuned SecBERT model weights, and the simulation scripts are publicly available at:

`https://github.com/Edu-p/traffic-light-secbert-br`

References

Bang, Y. et al. (2023). A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

Gupta, M., Akiri, C., Aryal, K., Parker, E., and Praharaj, L. (2023). From chatgpt to threatgpt: Impact of generative ai in cybersecurity and privacy. *IEEE Access*, 11:80218–80245. Open Access.

Inan, H., Upasani, K., Chi, J., et al. (2023). Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*.

Jiang, L., Hwang, J., Bhagavatula, C., et al. (2024). Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models. *arXiv preprint arXiv:2406.18510*.

Liu, Y. et al. (2023). Jailbreaking chatgpt via prompt engineering: An empirical study. *arXiv preprint arXiv:2305.13860*.

Presidência da República, Brasil (2018). Lei geral de proteção de dados pessoais (lgpd) – lei n° 13.709/2018.

Rababah, B. et al. (2024). Sok: Prompt hacking of large language models. *arXiv preprint arXiv:2410.13901*.

Rebedea, T., Dinu, R., Sreedhar, M., et al. (2023). Nemo guardrails: A toolkit for controllable and safe llm applications with programmable rails. arXiv preprint arXiv:2310.10501.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-based access control models. *Computer*, 29(2):38–47.

Souza, F., Nogueira, R., and Lotufo, R. (2020). Bertimbau: Pretrained bert models for brazilian portuguese. In *Lecture Notes in Computer Science*, pages 403–417. Springer.

Zhang, Z. and Jiexin, P. (2005). Delegation model for cscw based on rbac. In *Proc. International Conference on Network Computing and Information Security*, pages 367–371.