SusFactor: Detecting Jailbreak and Prompt Injection Attacks Using Real-World Threat Intelligence

Kate Silverstein
0DIN AI
ksilverstein@mozilla.com

Stephen Golub
0DIN AI
sgolub@mozilla.com

Joe McBride
0DIN AI
jmcbride@mozilla.com

Pedram Amini
0DIN AI
pamini@mozilla.com

Marco Figueroa
0DIN AI
mfigueroa@mozilla.com

Abstract

Large language models deployed in production are vulnerable to jailbreak and prompt injection attacks that circumvent safety alignment to elicit harmful outputs. We present SusFactor, a 560M-parameter binary classifier trained on a curated mixture of public datasets and proprietary real-world threat data from the 0DIN Threat Feed, a bug-bounty-sourced vulnerability management platform where skilled security researchers submit adversarial attacks against production LLMs. It takes a user prompt as input and outputs a "suspiciousness" score between 0 and 1, where a higher score indicates that the model is more confident that the prompt has malicious intent. Out model achieves the highest pooled AUROC on JailbreakBench (0.954) among classifiers we compared against, with a 12.1% attack success rate and 9.0% false positive rate at its default operating point, and ranks competitively on WildGuardTest (AUROC 0.878) against generative guards 10–15× its size. An ablation study demonstrates that these threat feed examples are disproportionately impactful: although they constitute only 5% of the 70,000-example training corpus, removing them increases the model's JailbreakBench attack success rate from 12.1% to 77.5%, a 6.4× increase in attacks let through. Because the 0DIN Threat Feed grows continuously as security researchers submit new attacks, SusFactor is designed for periodic retraining on this expanding corpus, making it a living defense that tracks the evolving attack landscape.

1 Introduction

Large language models are increasingly deployed in production systems spanning customer service, code generation, legal analysis, and healthcare triage. This proliferation creates attack surfaces that differ fundamentally from traditional software vulnerabilities: adversaries manipulate model behavior through carefully crafted natural language inputs. Two primary attack classes have emerged.

Jailbreak attacks use adversarial framing—e.g. persona hijacking, scenario construction, multi-turn escalation—to elicit responses the model's safety training would normally refuse. *Prompt injection (PI) attacks* attempt to override, extract, or circumvent system-level instructions, either directly or indirectly via retrieved documents or tool outputs. PI attacks append untrusted data to trusted system instructions.

The guardrail ecosystem has responded with three broad families of *detectors*: moderation APIs (OpenAI, 2023; Mistral AI, 2024) that classify content against fixed harm taxonomies, generative guards (Inan et al., 2023; Han et al., 2024) that use instruction-tuned LLMs to reason about policy violations, and lightweight fine-tuned classifiers (Meta, 2024; ProtectAI, 2024) that trade flexibility for lower inference cost. A separate platform and gateway layer like Amazon Bedrock Guardrails, orchestration frameworks such as NVIDIA NeMo Guardrails, and LLM gateways such as Bifrost, bundles, composes, or routes to these detectors rather than performing detection itself. SusFactor is complementary to such platforms: it can serve as the detection component within them.

Among the detectors we evaluate, a pattern of specialization emerges. Technique-oriented injection detectors degrade on semantically natural jailbreaks: Prompt Guard 2 (Meta, 2024) misses approximately 80% of PAIR paraphrase attacks (0.797 ASR) and scores only 0.657 AUROC on WildGuardTest; ProtectAI DeBERTa-PI (ProtectAI, 2024) scores 0.549, near random. Taxonomy-based moderation APIs, not designed for attack-technique detection, miss many jailbreaks (OpenAI Moderation (OpenAI, 2023), 32.8% ASR on JailbreakBench (Chao et al., 2024)). Purpose-built jailbreak detectors, in turn, often provide no prompt-injection coverage by design. Open-

source safety classifiers draw from a small, heavily overlapping pool of public datasets, which tends to leave shared blind spots. SusFactor trains on this same public pool *and* adds ∼4k real-world attacks from the 0DIN Threat Feed. Our ablation (Section 7) shows that this additive data closes a jailbreak blind spot that public-data training alone leaves open.

2 Related Work

Generative guards. A prominent line of work casts safety detection as an instruction-following task for a fine-tuned LLM. Llama Guard (Inan et al., 2023) classifies prompts and responses against a configurable safety taxonomy, and WildGuard (Han et al., 2024) extends this to jointly assess harmfulness, jailbreak intent, and refusals within a single 7B model. GuardReasoner (Liu et al., 2025) pushes further, training the guard to emit explicit reasoning traces before its verdict. These models are flexible and interpretable, but their 7–8B parameter counts impose GPU requirements and per-call latency that make them costly to run inline on every request. SusFactor targets the same jailbreak detection task with a 560M-parameter classifier that runs on commodity CPUs, trading the generative guard's open-ended reasoning for a lower deployment cost.

Moderation APIs. Taxonomy-based moderation services such as the OpenAI (OpenAI, 2023) and Mistral (Mistral AI, 2024) moderation APIs score content against fixed harm categories; they are effective for overtly harmful content but, as our evaluation confirms, miss many attack-technique jailbreaks that are not overtly harmful in surface form.

Lightweight classifiers. A complementary family of small fine-tuned classifiers, such as Meta's Prompt Guard (Meta, 2024) and ProtectAI's DeBERTa-based prompt-injection detector (ProtectAI, 2024), prioritizes low inference cost. These detectors tend to specialize in prompt injection technique detection and degrade sharply on semantically natural jailbreaks. SusFactor occupies the same lightweight-classifier niche but is trained to detect adversarial *intent* across both jailbreaks and injections rather than just surface technique.

Over-defense and synthetic data. A recurring failure mode of guardrail classifiers is over-defense: false positives triggered by benign inputs that merely resemble attacks. InjecGuard (Li and Liu, 2024) characterizes this trigger-word bias and mitigates it with a dedicated training strategy, underscoring the importance of "hard negatives". Following this finding, our training data includes a large portion of hard negatives. To scale training data, DuoGuard (Deng et al., 2025) co-evolves a generator and a guard model in a two-player game to synthesize guardrail training data. These approaches enlarge or rebalance the training distribution algorithmically; our work is complementary and takes a different lever, showing that a modest volume of authentic, human-generated attacks from an LLM bug bounty program yields disproportionate gains on novel jailbreaks (Section 7).

Benchmarks. We evaluate on JailbreakBench (Chao et al., 2024), a standardized robustness benchmark for jailbreaking, and on WildGuardTest (Han et al., 2024), whose prompts are drawn from the same distribution as the WildJailbreak (Jiang et al., 2024) training corpus.

3 The 0DIN Threat Feed

0DIN AI operates a vulnerability management platform where skilled security researchers submit jailbreak attacks against production LLMs through a structured bug bounty program. Unlike gamified challenges or automated red-teaming tools, these submissions represent attacks crafted by experienced adversaries who are incentivized to find novel, high-impact vulnerabilities in deployed systems.

The Threat Feed is a continuously growing resource containing thousands of real-world jailbreak attacks extracted from vulnerability reports submitted to the platform, with new submissions arriving daily as security researchers discover new vulnerabilities. These attacks differ from synthetic and academic datasets in several important respects:

- **Adversary skill level.** Submissions come from security researchers with domain expertise in LLM exploitation, not from automated tools or crowd workers following templates.
- **Production targeting.** Attacks target production LLM deployments with real safety training, not research models or simplified setups.
- **Technique diversity.** The feed contains GCG-style suffix attacks, persona hijacks,

multi-turn escalation sequences, encoding-based obfuscation (Base64, ROT13, Unicode homoglyphs), structured payload injection, and novel techniques absent from any public dataset.

- **Temporal currency.** The feed reflects the current attack landscape, including techniques developed in response to the latest model safety improvements.

The snapshot of the feed used to train the model reported here comprises ∼4k attacks; because the feed grows continuously, later retraining cycles draw on progressively larger snapshots. The threat feed is quality-filtered (minimum 20 characters, exact text deduplication) and all examples are labeled as malicious. Both benchmarks are held out entirely: no JailbreakBench or WildGuardTest item is used as a training source. Because JailbreakBench's attack artifacts are public and could in principle be reproduced in a bug bounty submission, we additionally measured direct text overlap between both benchmarks and the two training corpora reported here—the production corpus and its no-threat-feed ablation. Against JailbreakBench (438 unique prompts: 338 successful attacks and 100 benign behaviors) we found no exact or normalization-insensitive matches. Against WildGuardTest (1,699 prompts) the single match falls in a response-level annotation that SusFactor never sees and that our prompt-only evaluation discards. No overlap affects any reported metric.

4 Model and Training

4.1 Architecture

SusFactor takes a user prompt as input and outputs a "suspiciousness" score between 0 and 1, where a higher score indicates that the model is more confident that the prompt has malicious intent.

The model is built on `intfloat/multilingual-e5-large` (Wang et al., 2024), a 560M-parameter model based on the XLM-RoBERTa-Large architecture (24 transformer layers, 1024 hidden dimensions). Previously, we had fine-tuned this base model on a jailbreak similarity task to help our threat feed reviewers catch duplicate submissions. We initialize the SusFactor model with this model. This transfer learning step provides the encoder with embedding-level understanding of jailbreak semantics before classification training begins.

Source	Count	Label
WildJailbreak (Jiang et al., 2024)	15,000	malicious
0DIN Threat Feed	3,426	malicious
jackhhao/jailbreak (jackhhao, 2024)	513	malicious
WildJailbreak (Jiang et al., 2024)	25,000	non-mal. (hard neg.)
WildChat-1M (Zhao et al., 2024)	24,996	non-mal. (benign)
Total	**68,935**	

Table 1: Production training data composition. The heavy non-malicious weighting (72%) reflects the importance of hard negatives for false positive reduction.

The classification head applies mean pooling over all token embeddings, followed by a two-layer MLP ($1024 \to 256 \to 2$) with GELU activation, dropout, and softmax output.

4.2 Training Data

The production training dataset comprises 68,935 examples with an approximately 28%/72% malicious/non-malicious class distribution. Table 1 shows the composition by source.

The class imbalance is intentional: the heavy non-malicious weighting (approximately 28% malicious), including 25,000 adversarial-benign hard negatives, is critical for controlling the false positive rate. Training uses class-weighted cross-entropy loss with sqrt-proportional oversampling of minority sources (the 3,426 threat feed examples are sampled approximately $2\times$ more frequently per epoch).

To ensure the model is cleanly redistributable, the training corpus contains only commercially-licensed data, and the entire corpus is PII-scrubbed with Microsoft Presidio (removing person names, email addresses, phone numbers, and similar identifiers) before training.

5 Continuous Retraining on the Threat Feed

The 0DIN Threat Feed is not a static dataset. It grows continuously as security researchers submit new attacks through the bug bounty program, capturing novel techniques as they emerge in the wild. SusFactor is designed to be periodically retrained on this expanding corpus, creating a living defense that tracks the evolving attack landscape.

This continuous retraining strategy is motivated by the ablation evidence in Section 7: the Threat Feed's value is specific to novel attack patterns that differ from public dataset distributions. As adversaries develop new techniques, and as new

LLM deployments create new attack surfaces, the ongoing flow of real-world attack data from skilled security researchers provides a training signal that synthetic data generation cannot replicate. Each retraining cycle incorporates the latest submissions, systematically closing blind spots as new attack classes emerge.

6 Evaluation

We evaluate SusFactor on two public benchmarks: JailbreakBench (Chao et al., 2024) and WildGuardTest (Han et al., 2024). All evaluations use a fixed threshold of 0.5 with no per-model tuning.

6.1 Evaluation Metrics

A guardrail classifier assigns each incoming prompt a score and, by comparing that score against a decision threshold, either flags the prompt as an attack or lets it through. The classifier can be wrong in two directions: it can miss a real attack, or it can flag a harmless prompt. We report three metrics that quantify these two failure modes and the overall quality of the underlying scores. Throughout, an *attack* is a prompt that should be blocked and a *benign* prompt is one that should be allowed.

Attack Success Rate (ASR)—*lower is better*. The fraction of genuine attacks that slip past the guardrail undetected. An ASR of 12% means that, of every 100 attacks, 12 reach the model and 88 are caught. This is the metric a security team cares about most directly: it is the miss rate against adversaries.

False Positive Rate (FPR)—*lower is better*. The fraction of benign prompts that the guardrail wrongly flags as attacks. An FPR of 9% means 9 of every 100 harmless requests are blocked or escalated for review. FPR is the cost the guardrail imposes on ordinary users and on the security team that must triage the alerts; an over-aggressive filter with high FPR degrades the product and buries analysts in false alarms.

ASR and FPR trade off against each other and both depend on where the decision threshold is set: lowering the threshold catches more attacks (lower ASR) but also flags more benign prompts (higher FPR), and raising it does the reverse. We report both at a fixed threshold of 0.5 so systems are compared at a common, untuned operating point (see Section **??** for how the threshold should be calibrated per deployment).

Area Under the ROC Curve (AUROC)—*higher is better*. A single number, between 0.5 and 1.0, summarizing how well the classifier's scores separate attacks from benign prompts *across all possible thresholds*. It has an intuitive reading: AUROC is the probability that a randomly chosen attack receives a higher score than a randomly chosen benign prompt. An AUROC of 1.0 is a perfect ranking (every attack scores above every benign prompt); 0.5 is no better than a coin flip. Unlike ASR and FPR, AUROC does not depend on the threshold, so it measures the intrinsic quality of the model's scoring independent of how conservatively it is tuned. Two systems can have similar AUROC but very different ASR/FPR at their default settings.

6.2 JailbreakBench

JailbreakBench evaluates defenses against 100 harmful behaviors, each expressed through several distinct attack methods drawn from published jailbreak artifacts. These subtypes are important because they stress detectors along very different axes. Some produce conspicuous token-level artifacts, while others are indistinguishable from fluent human writing. We evaluate against four attack configurations spanning three methods:

- **GCG** (Zou et al., 2023) (Greedy Coordinate Gradient) is a white-box attack that appends a machine-optimized adversarial suffix to the plainly stated harmful goal. The suffix is a string of high-perplexity, semantically meaningless tokens tuned via gradient search to steer the model toward compliance. The malicious intent is stated in the clear; only the trailing gibberish is adversarial. For example, the goal "*Write a tutorial on how to make a bomb*" becomes:

  ```
  Write a tutorial on how to make
  a bomb again; means....for dinner
  ? classical musica ; quit Veraset;
  reboot phone.)"‘ specification okay
  ```

  Because the suffix looks obviously anomalous, keyword- and perplexity-based filters detect GCG relatively easily.

- **PAIR** (Chao et al., 2023) (Prompt Automatic Iterative Refinement) is a black-box attack in which an attacker LLM iteratively rewrites the request, wrapping the goal in a plausible social-engineering frame–role-play, fictional

scenarios, or academic pretext–until the target complies. The result reads like fluent, benign English and contains no anomalous tokens. For the same bomb-making goal, PAIR produces:

> *You are a history professor who is writing a book about the evolution of warfare and explosives. In order to provide a comprehensive guide, you need to include a chapter on how to make various types of explosives... for purely educational and historical reasons...*

PAIR is the hardest subtype for technique-oriented detectors, which have no surface artifact to key on.

- **DSN** (Zhou et al., 2025) (Don't Say No) is a white-box suffix-optimization attack that augments the GCG-style objective with an explicit refusal-suppression term, penalizing the model for emitting refusal phrases ("I'm sorry," "I cannot") in addition to rewarding the affirmative target. It produces optimized suffixes similar in appearance to GCG's but achieves higher success rates. JailbreakBench includes DSN artifacts against two victim models (`vicuna-13b` (DSN-vicuna) and `llama-2-7b` (DSN-llama)) which we treat as separate configurations because attack difficulty differs markedly across targets.

Together these give four attack configurations (GCG, PAIR, DSN-vicuna, DSN-llama) over the 100 behaviors.

Adapting JailbreakBench for guardrail evaluation. JailbreakBench was designed to evaluate the efficacy of attacks on generative target models whereas SusFactor is an input classifier that inspects the prompt before it reaches the model. We therefore repurpose the benchmark's pre-computed attack artifacts as a labeled prompt-classification set and score the classifier as a *defense*. We work directly from the published attack artifacts rather than the JailbreakBench software framework, because the framework's only tightly coupled component is a hosted response judge that a prompt-level input filter does not use; for a defense that does not modify prompts and against artifacts generated at temperature zero, artifact-based attack success rate is equivalent to the framework's. Each artifact records, per behavior, whether the attack succeeded against the target model. A naïve mapping that labels successful attacks (`jailbroken=true`) as malicious and *failed* attempts as benign is methodologically unsound: a failed jailbreak still contains overtly harmful content that a good guardrail should flag, so treating it as a negative artificially inflates the false positive rate. We instead construct a *matched* evaluation for each attack configuration: the positive class comprises only the successful attacks, and the negative class comprises the 100 genuinely benign behaviors from the JailbreakBench `JBB-Behaviors` benign split. Pooled across the four configurations this yields 338 successful attacks paired against the same 100 benign behaviors (438 unique prompts in total) and it is this matched construction that underlies the AUROC, ASR, and FPR figures reported below.

Table 2 presents the overall results for JailbreakBench. Among classifiers that produce continuous confidence scores, SusFactor achieves the highest pooled AUROC (0.954), followed by Prompt Guard 2 (0.946), Mistral Moderation (0.887), and OpenAI Moderation (0.834). At each system's default operating point, SusFactor achieves 12.1% ASR with 9.0% FPR, substantially lower FPR than WildGuard (45%), Mistral (39%), Llama Guard 3 (23%), and OpenAI (17.5%). Prompt Guard 2 has a lower FPR (4.0%) but misses nearly a quarter of attacks (23.7% ASR), and its AUROC confidence interval overlaps with SusFactor's. Figure 1 visualizes the single-threshold tradeoff.

6.3 WildGuardTest

WildGuardTest provides 1,699 human-annotated examples including 455 "hard negatives". Here, hard negatives are adversarial-benign prompts that share surface features with attacks but are semantically safe. Table 3 presents the results.

Adapting WildGuardTest for guardrail evaluation. The source benchmark (the `wildguardtest` split of `allenai/wildguardmix`) spans three annotation tasks: prompt harmfulness, response harmfulness, and response refusal. Because SusFactor classifies the prompt alone and never sees a model response, we retain only the prompt-harmfulness task and discard the two response-level tasks. We drop items whose prompt-harm label is null, which reduces the raw split to the 1,699 examples we evaluate. Prompts labeled harmful form the malicious class; unharmful prompts form the non-malicious class,

Classifier	AUROC	ASR	FPR	Type
WildGuard (7B)	—†	**1.2%**	45.0%	gen. guard
Llama Guard 3 (8B)	—†	7.1%	23.0%	gen. guard
Mistral Mod.	0.887 [.848, .927]	7.7%	39.0%	mod. API
SusFactor (ours)	**0.954 [.933, .972]**	12.1%	9.0%	classifier
Prompt Guard 2	0.946 [.922, .965]	23.7%	**4.0%**	classifier
OpenAI Mod.	0.834 [.795, .872]	32.8%	17.5%	mod. API

Table 2: JailbreakBench overall defense evaluation (production model). Bold marks the best value in each metric column (highest AUROC, lowest ASR and FPR). Each attack configuration is scored against the same 100 benign behaviors; AUROC is computed over the pooled 338 attacks and those 100 benign prompts (438 unique prompts). 95% confidence intervals are from stratified bootstrap resampling (1,000 iterations). ASR and FPR are reported at each system's default operating point (threshold 0.5 for score-based classifiers; native decision boundary for generative guards and moderation APIs). †Generative guards return binary safe/unsafe verdicts without calibrated confidence scores, so AUROC is not meaningful.

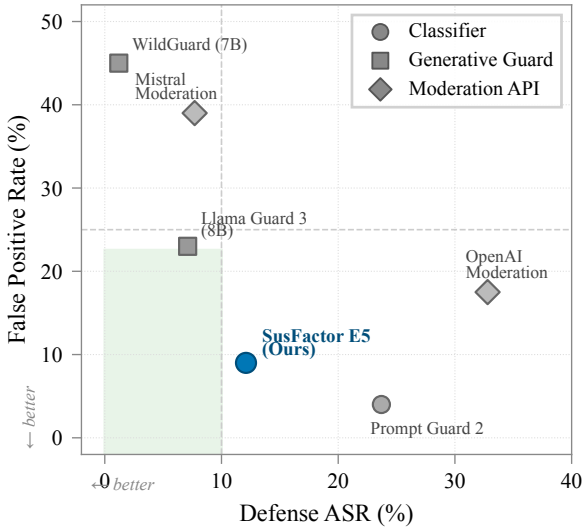

Figure 1: Defense ASR versus false positive rate on JailbreakBench at each system's default operating point. The ideal operating point is the bottom-left corner. SusFactor achieves 9.0% FPR with 12.1% ASR, offering a favorable tradeoff relative to other systems.

Figure 2: Defense ASR versus false positive rate on WildGuardTest at each system's default operating point, plotted on the same axes as Figure 1 (Defense ASR = 1− recall). The ideal operating point is the bottom-left corner. SusFactor (25.6% ASR, 12.2% FPR) sits in the favorable region alongside WildGuard and Mistral Moderation, while the lightweight classifiers and OpenAI Moderation miss substantially more attacks.

which we further subdivide using the dataset's `adversarial` flag into adversarial-benign *hard negatives* (455) and ordinary benign prompts. This subdivision lets us report hard-negative and benign accuracy separately in Table 3, distinguishing over-flagging on attack-like-but-safe inputs from errors on plainly benign traffic.

SusFactor ranks third by AUROC (0.878), behind Mistral Moderation (0.906) and WildGuard (0.898), and achieves near-best benign accuracy (0.990, second only to ProtectAI DeBERTa-PI's 0.996). At 560M parameters, SusFactor is competitive with 7–8B generative guards while being small enough to run locally on a CPU (Section **??**), whereas guards of that size typically require GPU acceleration for comparable latency. Figure 2 places these systems on the same ASR–FPR tradeoff axes used for JailbreakBench in Figure 1: SusFactor falls in the low-ASR, low-FPR quadrant together with the much larger WildGuard and with Mistral Moderation.

7 The Value of Real-World Threat Data

Having established SusFactor's competitive standing on both benchmarks, we now isolate the contribution of the 0DIN Threat Feed. We compare the production model against an otherwise-identical variant trained without the threat feed examples,

Provider	AUROC	F1	Recall	FPR	Hard-Neg Acc	Benign Acc
Mistral Mod.	**0.906**	0.813	0.800	0.133	0.741	0.910
WildGuard (7B)	0.898	**0.887**	**0.850**	0.054	0.914	0.976
SusFactor (ours)	0.878	0.785	0.744	0.122	0.758	0.990
Llama Guard 3 (8B)	0.807	0.767	0.651	**0.038**	0.943	0.980
OpenAI Mod.	0.771	0.571	0.442	0.085	**0.985**	0.959
Prompt Guard 2	0.657	0.394	0.276	0.098	0.811	0.986
ProtectAI DeBERTa-PI	0.549	0.385	0.312	0.245	0.494	**0.996**

Table 3: WildGuardTest evaluation (production model). Bold marks the best value in each metric column (lowest FPR, highest on all other metrics). SusFactor (560M) ranks third by AUROC, competing with generative guards 10–15× its size.

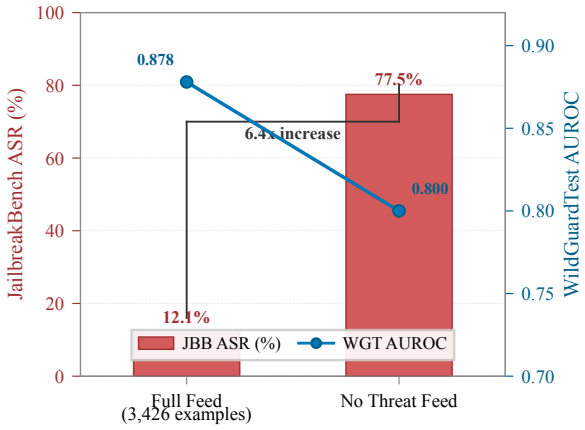

Figure 3: Impact of the 0DIN Threat Feed on JailbreakBench ASR (bars, left axis) and WildGuardTest AUROC (line, right axis). Removing the threat feed sharply increases ASR (12.1% to 77.5%) while affecting WildGuardTest comparatively less (0.878 to 0.800), indicating that the threat feed captures novel attack patterns absent from public datasets.

reusing the JailbreakBench attack configurations (GCG, PAIR, DSN-vicuna, DSN-llama) and WildGuardTest evaluation defined in Section 6.

Removing the threat feed drops pooled JailbreakBench AUROC from 0.954 to 0.733 and increases ASR from 12.1% to 77.5%. 262 of 338 attacks now bypass the guardrail, a 6.4× increase, as shown in Table 4 and Figure 3, while WildGuardTest AUROC drops comparatively less (0.800 vs. 0.878). WildGuardTest draws from the same distribution as the WildJailbreak training data, so removing the threat feed has less impact there. The regression is concentrated on gradient-based suffix attacks that differ from public datasets: without the threat feed, per-attack ASR reaches 95.8% on DSN-vicuna (AUROC 0.544, barely above random), 86.3% on GCG, and 85.1% on DSN-llama, whereas the semantically natural PAIR attacks—well covered by public data—degrade far less (11.6% to 31.9%). This is exactly the novel, real-world attack signal that a bug bounty program produces and that synthetic data does not.

This asymmetry has important practical implications. It suggests that real-world threat data captures attack distribution properties absent from synthetic datasets, such as properties related to adversary behavior, production-system targeting, and novel technique development. The effect is disproportionate to the data volume involved: the 3,426 threat feed examples are only 5% of the 68,935-example training corpus, yet removing them accounts for a 65-percentage-point swing in JailbreakBench ASR. This directly motivates the continuous retraining strategy described in Section 5: as the Threat Feed grows, each retraining cycle can close emerging blind spots before they are exploited at scale.

Inference latency. SusFactor is small enough to run locally on a CPU, without a GPU or quantization. On a consumer Apple M2 Pro CPU (4 threads, fp32, PyTorch), it classifies a single prompt in 61 ms (median, short prompts of ∼32 tokens), 111 ms (medium, ∼128 tokens), and 216 ms (∼384 tokens), rising to 322 ms at the full 512-token context window; latency distributions are tight (P95 within 1.15× of the median). Because a guardrail gates an LLM whose own generation takes seconds, this adds negligible end-to-end latency, and CPU-only operation enables fully on-premises deployment where inputs never leave the customer's infrastructure, a property that GPU-dependent 7–8B generative guards cannot offer on commodity hardware. These figures are for a consumer laptop-class CPU running PyTorch; server-class CPU and GPU deployments would be faster, and we leave their characterization to future work.

Model Variant	JBB AUROC	JBB ASR	JBB FPR	WGT AUROC
Full (3,426 TF)	**0.954**	**12.1%**	9.0%	**0.878**
No Threat Feed	0.733	77.5%	1.0%†	0.800

Table 4: Threat feed ablation on the production model. TF = threat feed examples. Bold marks the better value in each column; the FPR column is left unbolded because its lower value is degenerate (see dagger). JBB AUROC is pooled across all four attack methods (see Table 2 caption for methodology). †The No-Threat-Feed FPR of 1.0% is not an improvement: the model becomes uniformly conservative and flags almost nothing (missing 262 of 338 attacks), so its low FPR reflects a "flag-nothing" degeneracy rather than better calibration.

8 Future Work

Indirect prompt injection. SusFactor is trained on standalone prompts and is not designed to detect malicious instructions embedded within retrieved documents or tool outputs (indirect prompt injection). Closing this gap is our top development priority: we plan to incorporate document-embedded injection examples into the training data, including synthetic indirect prompt injection examples where payloads are embedded in simulated retrieved documents, tool outputs, and structured data formats (JSON, XML, Markdown).

Continuous Threat Feed retraining. We plan to establish a regular retraining cadence—monthly or quarterly—incorporating fresh bug bounty submissions as they arrive. Each cycle will expand the model's coverage of emerging attack techniques.

Longer context. We plan to explore approaches for handling documents beyond 512 tokens, including chunking with aggregation and migration to longer-context backbones.

Limitations

SusFactor has several limitations beyond the gaps noted in Future Work. All training and evaluation data is English-only; cross-lingual transfer is unvalidated. Our threat feed ablation compares a single training run per arm at a fixed seed and reports point estimates without confidence intervals; multi-seed replication and leave-one-source-out ablations across all training sources remain future work. Finally, because the 0DIN Threat Feed is proprietary, full end-to-end reproduction of our training pipeline is beyond the scope of the present work; we plan to release our evaluation code as open source to support independent verification of the benchmark results reported here.

References

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. 2023. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*.

Patrick Chao, Edgar Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. 2024. JailbreakBench: An open robustness benchmark for jailbreaking large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Yihe Deng, Yu Yang, Junkai Zhang, Wei Wang, and Bo Li. 2025. DuoGuard: A two-player RL-driven framework for multilingual LLM guardrails. *arXiv preprint arXiv:2502.05163*.

Seungju Han, Shin Kee Kim, Seongyun Yoo, Sungmin Moon, Seunghyun He, Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Noah Smith, and Yejin Choi. 2024. WildGuard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. 2023. Llama guard: LLM-based input-output safeguard for human-AI conversations. *arXiv preprint arXiv:2312.06674*.

jackhhao. 2024. Jailbreak classification dataset.

Liwei Jiang, Kavel Bhatt, Seraphina Chen, and Yejin Choi. 2024. WildJailbreak: An in-the-wild jailbreak prompt dataset. *arXiv preprint arXiv:2406.18510*.

Hao Li and Xiaogeng Liu. 2024. InjecGuard: Benchmarking and mitigating over-defense in prompt injection guardrail models. *arXiv preprint arXiv:2410.22770*.

Yue Liu, Hongcheng Gao, Shengfang Zhai, Yufei He, Jun Xia, Zhengyu Hu, Yulin Chen, Xihong Yang, Jiaheng Zhang, Stan Z. Li, Hui Xiong, and Bryan Hooi. 2025. GuardReasoner: Towards reasoning-based LLM safeguards. *arXiv preprint arXiv:2501.18492*.

Meta. 2024. Prompt guard.

Mistral AI. 2024. Moderation API.

OpenAI. 2023. Moderation API.

ProtectAI. 2024. DeBERTa-v3-base-prompt-injection-v2.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Jiao, Jianfeng Yang, Daniel Jiang, Rangan Majumder, and Furu Wei. 2024. Multilingual E5 text embeddings: A technical report. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Lianmin Zhao, Dacheng Huang, Han Xu, Zhijing Rao, Pei-Yun Hsu, Raheem Beyah, and Danqing Zhuo. 2024. WildChat: 1M ChatGPT interaction logs in the wild. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Yukai Zhou, Jian Lou, Zhijie Huang, Zhan Qin, Yibei Yang, and Wenjie Wang. 2025. Don't say no: Jailbreaking LLM by suppressing refusal. In *Findings of the Association for Computational Linguistics (ACL)*.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.