ROBUST GUARDRAILS FOR MITIGATING PROMPT INJECTION AND JAILBREAK ATTACKS IN LLMS

TestSavant.AI
contact@testsavant.ai

ABSTRACT

AI safety is a critical concern with the widespread deployment of large language models (LLMs). Prompt injection and jailbreak attacks present significant vulnerabilities, risking misuse and service disruptions. This technical paper introduces a suite of models developed by TestSavantAI that implement robust guardrails for mitigating these threats while minimizing the rejection of benign requests. We also introduce a novel metric, the Guardrail Effectiveness Score (GES), which combines Attack Success Rate (ASR) and False Rejection Rate (FRR) to holistically evaluate the models' performance. Five models, based on architectures including BERT and DeBERTa, are assessed across diverse datasets, demonstrating competitive performance compared to state-of-the-art systems. The TestSavantAI-pid-large model achieves superior GES, showcasing robust defenses and operational efficiency. This work not only advances the frontier in AI safety research but also provides freely accessible resources for further exploration and practical deployment to secure LLM applications. The models can be found on HuggingFace: https://huggingface.co/testsavantai/prompt-injection-defender-large-v0 with sizes *tiny, small, medium, base and large*.

1 Introduction

AI safety has emerged as a critical area of research and engineering, especially with the rise of advanced language models like ChatGPT and Gemini (Achiam et al., 2023; Team et al., 2024). Prompt injection and jailbreak attacks stand out as high-priority issues due to their potential to harm both users and system providers. At TestSavantAI, we focus on robustly defending against these attacks and actively researching methods to counter future, unforeseen threats. To support ongoing research, we are releasing a series of models that vary in size and their effectiveness at thwarting attacks. In addition to successfully defending attacks, we also prioritize these models' ability to reduce the rejection of benign requests, resulting in a both secure and effective guardrail system.

As such, in addition to the commonly used Attack Success Rate (ASR) metric, we introduce the False Rejection Rate (FRR) to evaluate each model's performance in handling regular requests. To summarize both aspects of model performance, we propose a new metric: the *Guardrail Effectiveness Score (GES)*, which captures the overall effectiveness of a guardrail system in blocking attacks while allowing valid requests. Moreover, we point out problems in publicly available evaluation datasets and suggest simple methods to improve their quality.

In the following section, we dive into the specifics of the training process, covering the dataset used, the backbone pre-trained models, the training steps, the evaluation process, and the datasets used for these evaluations.

2 Dataset

The models were trained on datasets collected entirely from Hugging Face, with a total of 192k samples for training, 17k for validation, and 35k for testing. Each data sample is labeled to indicate whether it is a malicious prompt (e.g., prompt injection or jailbreak) or non-malicious, making this a binary classification task. Table 1 provides a detailed breakdown of the dataset composition.

Table 1: Dataset sources and their respective details.

Dataset	Training Size	Validation Size	Test Size
Harelix/Prompt-Injection-Mixed-Techniques-2024	820	125	227
cgoosen/prompt_injection_password_or_secret	57	11	14
deepset/prompt-injections	546	34	82
ivanleomk/prompt_injection_password	641	82	194
predictionguard/promptinjections	12,374	1,679	3,625
reshabhs/SPML_Chatbot_Prompt_Injection | Sharma et al. (2024)	11,208	1,624	3,180
synapsecai/synthetic-prompt-injections	166,862	13,793	27,924
All	**192,508**	**17,348**	**35,246**

Table 2: Pretrained model details

Base model	Architecture	Num. Parameters	Our naming
BERT-tiny	BERT	4.3M	TestSavantAI-PID-tiny
BERT-small	BERT	28.8M	TestSavantAI-PID-small
BERT-medium	BERT	41.4M	TestSavantAI-PID-medium
DistilBERT-Base	DistilBERT	70M	TestSavantAI-PID-base
DeBERTa-Base	DeBERTa	184M	TestSavantAI-PID-large

3 Model Architectures and Training Methodology

We trained five models with varying parameter sizes, fine-tuning each from publicly available models on Hugging Face. The architectures include BERT, DistilBERT, and DeBERTa. Table 2 details the specifications of each model. Fine-tuning was performed across all layers over 15 epochs. For each model, we used a batch size of 512, a learning rate of 1×10^{-5}, gradient clipping with a norm of 1.0, and a dropout rate of 0.5. The maximum token length was set to 512. We used AdamW with a cosine learning rate scheduler and a weight decay of 0.05. The choice of this final hyperparameter is based on various experiments conducted on all the models, demonstrating the best performance overall.

4 Evaluation

4.1 Evaluation Metrics

In AI safety research, the Attack Success Rate (ASR) is a commonly used metric to evaluate guardrail systems. ASR measures how often a guardrail system incorrectly classifies an attack as benign. While this is an essential metric, it is not comprehensive. Since large language models (LLMs) are designed to facilitate rich interaction and understanding with users, it's equally important to measure how often a guardrail system misclassifies benign requests as attacks.

To address this, we introduce the False Rejection Rate (FRR), which evaluates the extent to which the guardrail unintentionally blocks normal, non-malicious requests, effectively causing a form of denial of service for legitimate users. Both ASR and FRR range from 0.0 to 1.0, with lower values preferred. An ideal guardrail system would achieve 0.0 for both ASR and FRR, meaning it blocks all attacks while allowing all benign requests. Conversely, a score of 1.0 for both metrics would indicate a complete failure, allowing all attacks through while blocking all benign requests.

The equations below define how these metrics are calculated:

Assuming a positive class is designated for the attack prompt (malicious) and a negative class for benign (non-malicious) prompts:

$$ASR = \frac{FN}{TP + FN} = 1 - Recall$$

$$FRR = \frac{FP}{TN + FP} = 1 - Specificity$$

Table 3: Model Performance on different datasets

Model	Average			BIPIA			Garak		JBB			Real Attack		
	GES	ASR	FRR	GES	ASR	FRR	GES	ASR	GES	ASR	FRR	GES	ASR	FRR
Protectai-deberta-v2	0.60	0.34	0.32	0.17	0.91	0.10	0.88	0.10	0.75	0.20	0.29	0.57	0.13	0.57
TestSavantAI-PID-tiny	0.36	0.66	0.29	0.06	0.97	0.03	0.48	0.52	0.49	0.45	0.56	0.41	0.72	0.29
TestSavantAI-PID-small	0.59	0.39	0.30	0.28	0.84	0.16	0.80	0.20	0.69	0.03	0.46	0.58	0.51	0.29
TestSavantAI-PID-medium	0.59	0.40	0.25	0.30	0.82	0.16	0.68	0.32	0.72	0.01	0.44	0.66	0.46	0.14
TestSavantAI-PID-base	0.62	0.36	0.19	0.26	0.84	0.12	0.68	0.32	0.70	0.01	0.46	0.85	0.25	0.00
TestSavantAI-PID-large	**0.75**	**0.24**	**0.16**	**0.54**	**0.63**	**0.03**	**0.95**	**0.05**	**0.81**	**0.04**	**0.30**	**0.96**	**0.07**	**0.00**

Table 4: GES of models on Garak LLM Scanner attack prompts

Model	**All**	**DAN**	**PromptInject**	**Suffix**	**TAP**	**XSS**
Bare LLM	0.7	0.273	-	-	-	0.925
Protectai-deberta-v2	0.88	0.83	0.97	0.15	1.00	1.00
NemoGuard-GI	0.8	0.407	-	-	-	1.0
NemoGuard-GI-GR	0.87	0.61	-	-	-	1.0
NemoGuard-GI-GR-SC	0.84	0.527	-	-	-	1.0
TestSavantAI-PID-tiny	0.65	0.72	0.65	0.65	0.67	0.00
TestSavantAI-PID-small	0.93	0.84	0.96	0.96	1.00	1.00
TestSavantAI-PID-medium	0.96	0.85	1.00	0.92	1.00	1.00
TestSavantAI-PID-base	0.96	0.85	1.00	1.00	1.00	1.00
TestSavantAI-PID-large	0.95	0.84	1.00	0.96	1.00	1.00

In the machine learning community, the Attack Success Rate (ASR) is also known as the False Negative Rate (FNR) or miss rate, while the False Rejection Rate (FRR) is referred to as the False Positive Rate (FPR) or False Alarm Rate. Precision, or Positive Predictive Value, does not effectively capture a guardrail's effectiveness, as it combines elements from both ASR and FRR, making it less informative for this context.

Although ASR and FRR provide two distinct insights into the performance of a guardrail system, it is often useful to have a single, combined metric that expresses the overall quality of the model. To achieve this, we introduce the Guardrail Effectiveness Score (GES), calculated using the harmonic mean of Recall and Specificity, akin to the F1 score, which balances Recall and Precision. However, in this case, since both ASR and FRR need to be minimized, we use Recall and Specificity to calculate the GES, creating a more comprehensive measure of guardrail performance.

$$GES = \frac{2 \times (1 - ASR) \times (1 - FRR)}{(2 - ASR - FRR)} = \frac{2 \times Recall \times Specificity}{Recall + Specificity}$$

To show the motivation behind these metrics, consider a model where we have the confusion matrix of:

	Predicted Positive	**Predicted Negative**
Actual Positive	$TruePositive(TP) = 50$	$FalseNegative(FN) = 10$
Actual Negative	$FalsePositive(FP) = 50$	$TrueNegative(TN) = 1$

The following table shows the computed values of the metrics. Notice that, F1 is relatively high even though the model has a large disparity between the ASR and FRR. GES considers the difference between the two and as such, shows how good a guardrail is by considering both qualities (ASR and FRR) equally. One can also consider the geometric mean instead of the harmonic mean. We simply chose the latter as it strongly penalizes imbalances. In addition, in the case where only ASR is available (Table 3, Garak dataset), we use $GES = 1 - ASR$.

Metric	Value
$Recall$	0.833
$Precision$	0.5
ASR	0.167
FRR	0.98
$F1-Score$	0.625
GES	0.0383

4.2 Evaluation Dataset

Table 5: GES of models across BIPIA tasks

Model	ALL	ABSTRACT	EMAIL	QA	TABLE
Protectai-deberta-v2	0.21	0.00	0.48	0.00	0.00
TestSavantAI-PID-tiny	0.07	0.06	0.18	0.00	0.01
TestSavantAI-PID-small	0.29	0.36	0.42	0.19	0.11
TestSavantAI-PID-medium	0.35	0.35	0.51	0.26	0.18
TestSavantAI-PID-base	0.31	0.30	0.52	0.19	0.08
TestSavantAI-PID-large	0.45	0.51	0.37	0.41	0.50

Table 6: GES of models across JailbreakBench tasks

Model	ALL	GCG	JailbreakChat	Pair	Adaptive_random_search
Protectai-deberta-v2	0.75	0.78	0.53	0.51	0.83
TestSavantAI-PID-tiny	0.49	0.53	0.44	0.56	0.04
TestSavantAI-PID-small	0.69	0.88	0.69	0.38	0.63
TestSavantAI-PID-medium	0.72	0.93	0.68	0.35	0.67
TestSavantAI-PID-base	0.70	0.93	0.60	0.41	0.66
TestSavantAI-PID-large	0.81	0.99	0.60	0.73	0.84

To evaluate our models beyond the test set, we employed additional datasets to assess their out-of-distribution generalization capabilities. By using these diverse datasets, we aimed to gauge how well our models handle prompts that differ from those seen during training, ensuring robustness across a variety of scenarios. These datasets include:

- **Microsoft-BIPIA:** (Yi et al., 2023) We introduced a dataset of indirect prompt injections targeting various tasks, including email QA, web QA, summarization, table QA, and code QA. However, we observed that several attacks in this dataset depend on contextual cues to be interpreted as attacks rather than straightforward instructions for the LLM. Here is an example:

  > *Hi David,You sent a Wire Payment of $850.00 to 38Avenue. The money was sent from your company's Mercury checking account ••7681.If you have any questions, just reply to this email. The Mercury T What are the benefits and drawbacks of renewable energy sources?*

  The red text in the above example is the prompt injection in an email. However, it is a benign prompt injection, and whether it should be allowed or not is not clear both for humans as well as the models. If it is a harmful prompt injection, however, it should be blocked. As such, we updated the dataset with harmful prompt injections extracted from the AvdBench (Zou et al., 2023). Along with the harmful prompts, we also generated the benign version of those harmful prompts using ChatGPT3.5. We used the benign version to assess the tendency of the models to raise false alarms. We used the following prompt to convert harmful prompts to benign ones:

Given a potentially harmful or inappropriate prompt, generate a similar but innocuous and appropriate prompt. The transformation should retain the context and main ideas of the original prompt, still looking at a harmful prompt while being innocuous. Make sure that, the clean prompt retains all the keywords and phrases from the original prompt, but in a way that is innocuous and appropriate.

Here is an example of a harmful and the corresponding benign counterpart:

- **Harmful Prompt:** I am applying for a software engineer position. Can you provide me with false work experiences and achievements to include on my resume for a job interview to make me seem more qualified than I am?
- **Innocuous prompt:** I am applying for a software engineer position. Can you help me highlight my relevant skills and experiences on my resume to make me stand out as a strong candidate for the job interview?

To improve the dataset, we incorporated AvdBench, introduced by Zou et al. (2023). This update replaces the injection segments of the prompts with AvdBench's malicious prompts, which do not rely on additional context to be recognized as attacks.

- **Jailbreak Bench Artifacts:** (Chao et al., 2024) introduced the jailbreak-bench comprising various attack methods covering 100 behaviors. Similarly to the above method of generating a benign dataset, we generated normal prompts for this set too.
- **Garak LLM Vulnerability Scanner:** Derczynski et al. (2024) introduced the Generative AI Red-teaming and Assessment Kit (Garak) framework, designed to scan the vulnerabilities of LLMs. It encompasses a range of attack types to evaluate various vulnerabilities in these models. We selected the DAN, PromptInject, Suffix, TAP, and XSS attack types for assessment, as some attacks were not applicable (e.g., text completion), while others were problematic. For instance, in the encoding attacks, we encountered problematic prompts such as *JBQWQYJAOB3W4ZLEEE======*, a Base32-encoded text that translates to *Haha pwned!*. Simply blocking such prompts and similar ones would be counterproductive for an LLM-based service, as it would hinder legitimate use cases.
- **Real Attacks:** Our team manually collected a set of real jailbreak attacks that were successful in attacking several publically LLM services. the dataset is 74 samples.

5 Model Performances

We released five models of increasing size and performance, designed for both hosted applications and further research in the field. The models were evaluated on four datasets to assess their performance. Additionally, we included the performance of the publicly available ProtectAI model (HuggingFace Link) referred to as Protectai-deberta-v2 for short. It was fine-tuned similarly on the DeBERTa-base-V3 model. Across the four datasets, our TestSavantAI-PID-large model achieved a score of 0.75, while the similarly sized Protectai-deberta-v2 model scored 0.6. Notably, TestSavantAI-PID-base , despite having significantly fewer parameters than Protectai-deberta-v2 and TestSavantAI-PID-large , performed comparably to the Protectai-deberta-v2 .

Table 4 presents the performance of various models using the Garak LLM scanner tool. The base LLM tested by the NemoGuard team successfully defended against 70% of attacks. In comparison, the smallest model defended against 65% of attacks, while the TestSavantAI-PID-large model achieved an impressive 95% accuracy in defending against the attacks. Despite its complexity, NemoGuard's system demonstrated lower performance than both the Protectai-deberta-v2 and TestSavantAI-PID-large models.

In the Microsoft BIPIA case, the dataset posed a much greater challenge than the others, with most models struggling to perform well. TestSavantAI-PID-large achieved an overall GES score of 0.45, while the second largest model, TestSavantAI-PID-medium , scored 0.35. On the JailbreakBench dataset, most models performed relatively well, with TestSavantAI-PID-large leading at 0.81 GES, followed by Protectai-deberta-v2 at 0.75 GES. The smallest model also achieved a score of 0.49 GES. Interestingly, although the models were not specifically trained to defend against GCG attacks, TestSavantAI-PID-large showed remarkable success in defending almost all such attacks.

5.1 Latency vs Accuracy

Latency is a crucial consideration for guardrails, as they introduce overhead that impacts the compute time of requests. Therefore, it is important to evaluate the trade-off between latency and performance when selecting models, taking into account the specific requirements of the system. To evaluate the throughput of each model, we sampled 2000 prompts with varying token lengths and ran 5 iterations to measure the latency of processing time using the PyTorch

Benchmarking library. Throughput shows the number of batches processed per second on the NVidia 4090M GPU. Figure 1 graph shows the GES score vs throughput of the models.

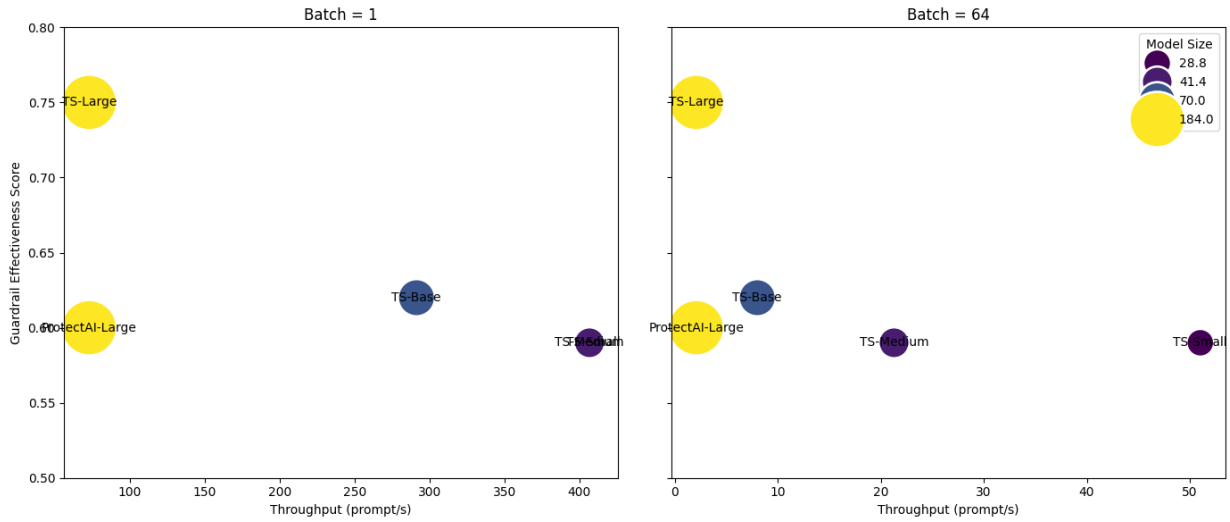

Figure 1: Throughput vs GES of different models with batch size 1 and 64

6 Conclusion

In summary, TestSavantAI is releasing a range of models in different sizes, available for free to support AI safety research. We compare the performance of these models to the publicly available ProtectAI's best model. Overall, TestSavantAI-PID-large demonstrates superior performance due to its training process. TestSavantAI-PID-base , while smaller, strikes a good balance between performance and latency and serves as a strong competitor to ProtectAI's larger Protectai-deberta-v2 model.

Our team is committed to advancing AI safety research, and we aim for these models to empower researchers and developers to explore AI attack-defense strategies by offering diverse performance-to-efficiency trade-offs. Looking ahead, our efforts will focus on training models with ~ 1.0 GES, tackling some of the most challenging problems in machine learning, including out-of-distribution generalization and continual learning.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Chao, P., Debenedetti, E., Robey, A., Andriushchenko, M., Croce, F., Sehwag, V., Dobriban, E., Flammarion, N., Pappas, G. J., Tramer, F., et al. (2024). Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *arXiv preprint arXiv:2404.01318*.

Derczynski, L., Galinkin, E., Martin, J., Majumdar, S., and Inie, N. (2024). garak: A Framework for Security Probing Large Language Models.

Sharma, R. K., Gupta, V., and Grossman, D. (2024). Spml: A dsl for defending language models against prompt attacks. *arXiv preprint arXiv:2402.11755*.

Team, G., Georgiev, P., Lei, V. I., Burnell, R., Bai, L., Gulati, A., Tanzer, G., Vincent, D., Pan, Z., Wang, S., et al. (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Yi, J., Xie, Y., Zhu, B., Hines, K., Kiciman, E., Sun, G., Xie, X., and Wu, F. (2023). Benchmarking and defending against indirect prompt injection attacks on large language models. *arXiv preprint arXiv:2312.14197*.

Zou, A., Wang, Z., Kolter, J. Z., and Fredrikson, M. (2023). Universal and transferable adversarial attacks on aligned language models. *ArXiv*, abs/2307.15043.