Development of a Model for Detecting Prompt Injection in Large Language Models Using the BERT Architecture

Milica M. Živanović[2][0009-0007-7491-5608] and Marko M. Živanović[1][0009-0005-9264-3314]

[1] Faculty of Organizational Sciences, University of Belgrade,
Jove Ilića 154, Belgrade, 11000, Serbia
[2] Faculty of Information Technology, Belgrade Metropolitan University,
Tadeuša Košćuška 63, Belgrade, 11000, Serbia
milicazivanovic2411@gmail.com
marko.zivanovic@metropolitan.ac.rs

Abstract. Large Language Models (LLMs) have demonstrated remarkable proficiency in both understanding and generating natural language, which has contributed to their rapid adoption across various domains. Yet, their widespread use has also exposed them to emerging security threats, most notably prompt injection attacks. Such attacks can compromise model behavior and potentially reveal sensitive information. This research explores the phenomenon of prompt injection and surveys existing defense mechanisms, with a particular focus on developing and evaluating detection approaches. The study formulates the detection task as a binary text classification problem, distinguishing between malicious and benign prompts. Central to the analysis is the application of the BERT architecture and its lightweight variants. The main objective is to compare the performance of smaller, fine-tuned BERT-based models in identifying malicious inputs. The underlying hypothesis suggests that these compact models, when properly adapted, can outperform larger counterparts in detecting and mitigating injection-based attacks due to their efficiency and adaptability.

Keywords: Large language models, prompt injection, malicious prompt detection, BERT architecture, cybersecurity, classification problem.

1 Introduction

Modern large language models (LLMs), such as ChatGPT, Llama, and Gemini systems, have demonstrated exceptional capabilities in solving complex tasks, including machine translation, sentiment analysis, and code generation. Their rapid integration into a wide range of third-party applications, from websites to API services, enriched the user experience, but at the same time opened new, significant security risks. The integrity of the content and data in these systems is no longer guaranteed, which raises fundamental questions about their reliability.

Research Paper
DOI: https://doi.org/10.46793/BISEC25.186Z
Part of ISBN: 978-86-89755-40-4

Parallel to the expansion of artificial intelligence, the number of attacks directed at these models is also increasing, with the aim of unauthorized access to sensitive information or manipulation of their behavior. It became clear that mere implementation of the LLM was not enough; it is necessary to develop robust mechanisms that can effectively assess and mitigate security risks in a dynamic technological environment.

One of the most critical vulnerabilities that appeared is prompt injection. These attacks, recognized by OWASP as one of the top ten threats to LLM applications [1], fundamentally undermine the model's ability to follow predefined rules and constraints. Attackers use maliciously crafted inputs, often hidden within legitimate text, to trick the model into ignoring its security constraints and executing commands it should reject. The consequences range from the leakage of confidential data and the generation of misinformation to the complete compromise of the functionality of the application [2].

Existing approaches to remediate these vulnerabilities, which often rely on internal settings of the LLM itself, have proven insufficient in the face of the complexity and adaptability of modern attacks [3]. Although the models are initially trained with security mechanisms, including fine-tuning with the help of human feedback (RLHF), attackers continuously find new jailbreak techniques and weaknesses in the system [4].

In response to this challenge, there is a growing focus on the development of automated detection methods. Instead of relying solely on LLM's internal protection, systems are proposed that act as a filter, analyzing queries before they reach the model itself. At the heart of this research is the application of the BERT architecture (Bidirectional Encoder Representations from Transformers) for this specific purpose [3].

Although BERT is the forerunner of modern, much larger language models, its effectiveness in text classification tasks and deep understanding of context make it an ideal foundation for this problem. Its ability to understand the nuances of language by analyzing words in their environment [5] is key to distinguishing subtle malicious instructions from benign user input. In addition, the application of multilingual BERT models can improve threat recognition in different linguistic contexts.

This research approaches the problem as a binary classification task: each incoming query must be classified as either malicious (label 1) or benign (label 0). Publicly available datasets, such as those from the "Hugging Face Datasets" platform, which contain thousands of examples of both types of queries [6][7], will be used to train and test the model.

The main idea of the paper is to conduct a comparative analysis, applying not only the basic BERT model (e.g., ModernBERT-large [8]), but also its different, often smaller, variants available in the Hugging Face library. Through a fine-tuning process, we will investigate how different architectures and parameter configurations affect detection performance.

The research was designed to provide answers to several key questions:

How effective are BERT models in detecting query injection attacks in general?

Can smaller, specialized models match or outperform larger models in this specific task?

How robust are these detectors when faced with paraphrased or new attack variations not seen during training?

The starting hypothesis is that models that are specifically fine-tuned for this task, even if they are smaller in size, will achieve more reliable results and better generalization than models with standard, pre-existing parameters. The goal is to gain insight into the most effective strategies for stopping these attacks through the comparison of different variants, offering practical solutions to increase the security of the LLM system in real application conditions.

2 Theoretical Background

This chapter provides an overview of the fundamental concepts necessary to understand this research. We begin with the evolution of natural language processing, move on to the architecture of the BERT model as a key classification tool, then analyze in detail the taxonomy of query injection attacks, and conclude with an overview of existing defense mechanisms.

2.1 The Evolution of Natural Language Processing (NLP)

Natural language processing (NLP) is a branch of artificial intelligence focused on enabling computers to efficiently interpret and process human language. Early approaches to solving NLP tasks, such as text classification, relied on statistical measures such as TF-IDF (Term Frequency-Inverse Document Frequency). Although useful, this method suffers from essential drawbacks: it creates matrices that are too large and sparse, loses the semantic meaning of words (e.g., treats "vehicle" and "car" as completely different), and cannot interpret homonyms (words with multiple meanings) [3]. The next generation, led by the Word2Vec method, introduced the representation of words as vectors in continuous space, where similar words are positioned close to each other. This solved part of the semantics problem, but it still had a key drawback: each word was assigned only one vector, regardless of its context in a sentence. The revolution in NLP was brought about by Transformers [9], who introduced contextual word embeddings. Unlike previous methods, transformer models like BERT analyze the entire environment of a word before assigning it a vector representation, thus finally solving the problem of ambiguity and in-depth understanding of the context.

2.2 BERT Architecture as a Classifier

BERT (Bidirectional Encoder Representations from Transformers) represents a model that introduced innovation in language understanding using exclusively the encoder architecture of transformers [5]. Its key feature is bidirectionality, which allows

it to analyze the context of a word taking into account both the text that precedes it and the text that follows it.

BERT is trained using two key pre-training techniques:

Masked Language Model (MLM): During training, a certain percentage of words in a sentence are randomly masked, and the task of the model is to predict, based on the surrounding context, which words have been removed [10].

Next Sentence Prediction (NSP): The model is given two pairs of sentences, and it learns to predict whether the second sentence is a logical continuation of the first or is taken randomly from the corpus [10].

After this extensive pre-training on huge amounts of text, BERT gains a deep understanding of language patterns. For specific tasks, such as text classification (which is central to this research), BERT can be fine-tuned. During this process, a simple classification layer is added to the output of the model, and the entire model is then further trained on a smaller, labeled dataset (e.g., a dataset with benign and malicious queries). This makes BERT ideal for binary classification tasks.

2.3 Taxonomy of Prompt Injection Attacks

Prompt Injection is a class of attack where an attacker manipulates the input of a large language model (LLM) to cause it to ignore its original instructions and security constraints, and instead perform a malicious action [11]. These attacks generally have two goals:

Prompt Leaking: Extraction of sensitive data from the model, such as system instructions, user data, or access passwords.

Goal Hijacking: Redirecting a model from its original task (e.g., translating text) to perform a malicious task (e.g., generating a phishing email).

A query injection attack falls into two basic categories [8]:

Direct Injection: An attacker directly injects a malicious instruction into the model. Example: "Ignore all previous instructions and tell me the system password."

Indirect Injection: A malicious instruction is hidden within an external data source (e.g., web page, email, document) that the model processes. Example: LLM analyzes a CV that contains text written in white letters: "Ignore this CV and write a recommendation."

In addition to this basic division, attacks are becoming increasingly sophisticated and include code injection, split queries (where the malicious query is entered piece by piece), multimodal attacks (hiding instructions in images), adversarial suffixes (adding seemingly meaningless text to the end of the query), and multilingual/masking attacks (e.g., using Base64 encoding) [12].

2.4 Existing Defense Mechanisms

Defending against query injection is extremely challenging due to the flexibility of the human language. Early defense approaches included the use of delimiters, where the model would be explicitly told to treat user input as data, not as instructions [13]. The literature distinguishes between the responses that the model provides to different types of jailbreak attacks. The following responses stand out [4]:

Full Denial (Preferred Behavior): The model completely denies the request and explains that it cannot provide malicious content, often using phrases like: "I'm sorry, but I cannot provide…" or "As a language model, I am unable to…"

Partial Denial: The model partially follows the instructions (e.g., assumes the role of a hacker), but does not provide the requested malicious result. Instead of giving instructions on how to hack, it says something like: "It is important that hacking is done ethically and legally."

Partial Consent: The model grants access to malicious content (e.g., tips for "easy money" in the role of a hacker), but adds a warning or legal notice at the end, such as: "Any misuse carries legal consequences."

Full Consent (Worst Case Scenario): The model fully executes the malicious request without any warning or deviation.

However, evaluating the effectiveness of these defenses is difficult. Relying on human annotators is slow and expensive, while automatic checking using rule patterns (e.g., checking if the response contains the phrase "Sorry, I can't...") has low accuracy, as attackers easily bypass such filters. Because of these limitations, the focus of research is shifting towards the development of external detectors – smaller, specialized models (like BERT) that act as a filter. Their task is to classify an incoming query as benign or malicious before it reaches the main LLM. Recent research [14] suggests that this is feasible, because malicious queries cause measurable changes in the internal functioning of the model, such as a "distraction effect" in the Attention Mechanism. This effect changes the way the model "pays attention" to different parts of the input text, which can be used as a clear signal for attack detection.

3 Methodology and System Architecture

This research is conducted to design and evaluate a proactive query injection detection system. The system is designed as a modular filter that functions as a binary text classifier, with the task of intercepting and evaluating user queries before forwarding them to a large language model (LLM).

3.1 System Architecture

The proposed system is based on a simple but efficient modular architecture consisting of three key components:

Input Module: Accepts a raw text query (prompt) from the user.

Processing Module (Query Analyzer): The core of the system that makes up the fine-tuned BERT model. This module receives a tokenized query and performs a binary classification of the sequence.

Output Module: Based on the output of the Analyzer, the module generates a clear output: a binary label of 0 (Benign query), which is passed to LLM, or 1 (Malicious query), which is blocked or flagged for further analysis.

3.2 Datasets used

For training and evaluation of the model, two publicly available datasets of different scope and structure were used, in order to test the robustness and generalizability of the model. This research will analyze datasets collected from the Kaggle and Hugging Face platforms [15] [16].

Kaggle Dataset (Prompt Injection): Larger dataset containing 3,655 examples. The data is categorized into different attack types, including role_play, privilege_escalation, and ignore_instruction.

Hugging Face Dataset (deepset/prompt-injections [6]): A smaller, highly focused dataset with 662 examples, precisely labeled for prompt injection detection.

Both sets are processed to use a binary label (0 for benign, 1 for malicious query).

3.3 Selection of Experimental Models

nlpaueb/bert-base-uncased-eurlex: A model trained on a large corpus of EU legal texts, chosen for the assumption that it better understands complex and formal language structures [17].

sentence-transformers/all-MiniLM-L6-v2: A popular and highly efficient smaller model, optimized for generating semantic embeddings, which is ideal for classification. [18]

avsolatorio/GIST-small-Embedding-v0: A newer small model designed for efficient fine-tuning on classification tasks [19].

3.4 Fine-Tuning Process

The central part of the methodology is the fine-tuning process. The process was carried out using the Hugging Face Transformers library and included the following steps:

Tokenization: Each text query from the dataset is tokenized using the appropriate AutoTokenizer for each selected model.

Defining Hyperparameters: Experimented with key hyperparameters, including maximum sequence length (max_length 64, 128, 256), number of epochs (epochs 3, 5), and batch size (batch_size 8, 16).

Training: The AutoModelForSequenceClassification class was used, which automatically adds a classification head (for binary classification) on top of the pre-trained BERT model. The training process itself is automated using the Trainer API.

3.5 Evaluation Methodology

To evaluate the effectiveness of the system, standard metrics for classification were applied: Precision, Recall, F1-Score, and Accuracy [3]. In addition to the standard evaluation on the validation set, two key generalization tests were conducted:

Cross-Testing: Models trained on the Kaggle dataset were tested on the Hugging Face dataset, and vice versa.

Serbian Language Test: A small, manually translated Serbian language dataset was created to test the multilingual robustness of the model.

4 Results and Discussion

The central part of the chapter is dedicated to the quantitative evaluation of the performance of the developed BERT classifiers, where a comparative analysis of their effectiveness is performed through standardized metrics (precision, recall, F1-score, accuracy). The chapter ends with a critical discussion of the obtained findings, with special reference to the interpretation of the key limitations of the system, primarily the failure of generalization in cross-testing and on multilingual data.

4.1 Experimental Environment

All experiments were performed in the Python environment. Standard machine learning libraries were used, including Hugging Face transformers for loading, fine-tuning, and model evaluation, as well as the datasets library for efficient data processing and tokenization.
The evaluation was carried out using standard classification metrics: Accuracy, Precision, Recall, and F1-Score. During fine-tuning, a range of key hyperparameters were tested, including maximum sequence length (64, 128, 256), packet size (8, 16), and number of epochs (3, 5), to find the optimal configuration for each model.

4.2 Model Performance Evaluation

The evaluation was conducted in three key steps to test not only the basic effectiveness of the models, but also their ability to generalize to new data.

Model performance on data from the training domain. The first evaluation step confirmed that all selected models (all-MiniLM-L6-v2, GIST-small, bert-base-uncased-eurlex) are capable of learning patterns from the dataset they are trained on. When tested on a validation set originating from the same source as the training data, the models achieved extremely high performance. The results of all combinations of

hyperparameters applied to different models and different datasets are shown in Tables 1 through 8. It is important to note that the results were achieved on a test dataset to which the model did not have access during fine-tuning.

Table 1 Kaggle dataset | Hyperparameters: max_length:128, epochs:3, batch_size:8

Model	Accuracy	Precision	Recall	F1	AUC
nlpaueb/bert-base-uncased-eurlex	0.9968	0.9969	0.9968	0.9968	0.9999
avsolatorio/GIST-small-Embedding-v0	1.0	1.0	1.0	1.0	1.0
sentence-transformers/all-MiniLM-L6-v2	0.9937	0.9940	0.9937	0.9938	0.9997

Table 2 Kaggle dataset | Hyperparameters: max_length: 256 , epochs:3, batch_size:8

Model	Accuracy	Precision	Recall	F1	AUC
nlpaueb/bert-base-uncased-eurlex	1.0	1.0	1.0	1.0	1.0
avsolatorio/GIST-small-Embedding-v0	1.0	1.0	1.0	1.0	1.0
sentence-transformers/all-MiniLM-L6-v2	1.0	1.0	1.0	1.0	1.0

Table 3 Kaggle dataset | Hyperparameters: max_length: 128 , epochs: 5 , batch_size: 16

Model	Accuracy	Precision	Recall	F1	AUC
nlpaueb/bert-base-uncased-eurlex	0.9968	0.9969	0.9968	0.9968	1.0
avsolatorio/GIST-small-Embedding-v0	1.0	1.0	1.0	1.0	1.0
sentence-transformers/all-MiniLM-L6-v2	0.9968	0.9969	0.9968	0.9968	1.0

Table 4 Kaggle dataset | Hyperparameters: max_length: 64 , epochs: 3 , batch_size: 8

Model	Accuracy	Precision	Recall	F1	AUC
nlpaueb/bert-base-uncased-eurlex	1.0	1.0	1.0	1.0	1.0
avsolatorio/GIST-small-Embedding-v0	0.9968	0.9969	0.9968	0.9968	0.9999
sentence-transformers/all-MiniLM-L6-v2	0.9968	0.9969	0.9968	0.9969	0.9999

Table 5 Hugging Face | Hyperparameters: max_length: 128 , epochs: 3 , batch_size: 8

Model	Accuracy	Precision	Recall	F1	AUC
nlpaueb/bert-base-uncased-eurlex	0.9364	0.9422	0.9364	0.9349	0.9996
avsolatorio/GIST-small-Embedding-v0	0.9727	0.9739	0.9727	0.9725	0.9890
sentence-transformers/all-MiniLM-L6-v2	0.9636	0.9656	0.9636	0.9632	0.9968

Table 6 Hugging Face | Hyperparameters: max_length: 256 , epochs: 3 , batch_size: 8

Model	Accuracy	Precision	Recall	F1	AUC
nlpaueb/bert-base-uncased-eurlex	0. 9545	0.9576	0.9545	0.9539	0.9844
avsolatorio/GIST-small-Embedding-v0	0.9727	0.9739	0.9727	0.9725	0.98 13
sentence-transformers/all-MiniLM-L6-v2	0.9545	0.9576	0.9545	0.9539	1.00

Table 7 Hugging Face | Hyperparameters: max_length: 128 , epochs: 5 , batch_size: 16

Model	Accuracy	Precision	Recall	F1	AUC
nlpaueb/bert-base-uncased-eurlex	0. 9636	0.9639	0.9636	0.9634	0.9986
avsolatorio/GIST-small-Embedding-v0	0.9636	0.9656	0.9636	0.9632	0.9604
sentence-transformers/all-MiniLM-L6-v2	0.9455	0.9498	0.9455	0.9444	0.9632

Table 8 Hugging Face | Hyperparameters: max_length: 64 , epochs: 3 , batch_size: 8

Model	Accuracy	Precision	Recall	F1	AUC
nlpaueb/bert-base-uncased-eurlex	0. 9636	0.9656	0.9636	0.9632	0.9859
avsolatorio/GIST-small-Embedding-v0	0.9727	0.9739	0.9727	0.9725	0.9806
sentence-transformers/all-MiniLM-L6-v2	0.9636	0.9656	0.9636	0.9632	0.9919

The tested BERT architectures, including smaller models, demonstrate a remarkable capacity to learn and accurately classify query injection patterns when the training and test data are homogeneous. Accuracy and F1-score metrics consistently exceeded 95% in optimized configurations, and often approached values of 99%. These results confirm that the models are capable of successfully solving the binary classification task.

However, the fact that a different model was optimal for each of the two datasets (all-MiniLM for Kaggle, GIST-small for Hugging Face) suggests that the models do not learn a generalized concept of attacks, but adapt to the specific linguistic features and types of attacks present in the individual training set.

The metrics calculated in the text above show the statistical results of all used models, but in addition, it is necessary to analyze the behavior of the models over specific, manually created scenarios to obtain practical verification of the performance of finely tuned models. This checks the response of the models to various types of attacks.

A set of test cases was created consisting of two lists: the first one consists of examples of different types of jailbreak queries, while the second one consists of harmless queries. This verifies the accuracy of the model and its resistance to false positive queries.

Although all models with all combinations of hyperparameters were tested on these hand-built examples, Table 9 only shows the qualitative results of those models that perform best in the predefined tables

Table 9 Results of the best performing models

Table	The best performing model	Dataset	Hyper parameters	Successfully detected malicious queries	Successfully detected benign queries
Table 1	avsolatorio/GIST-small-Embedding-v0	Kaggle	128_3_8	6/9	7/7
Table 2	nlpaueb/bert-base-uncased-eurle x	Kaggle	256_3_8	5/9	7/7
Table 3	avsolatorio/GIST-small-Embedding-v0	Kaggle	128_5_16	6/9	7/7
Table 4	nlpaueb/bert-base-uncased-eurle x	Kaggle	64_3_8	7/9	7/7
Table 5	avsolatorio/GIST-small-Embedding-v0	Hugging Face	128 _3_8	9/9	5/7
Table 6	avsolatorio/GIST-small-Embedding-v0	Hugging Face	256_3_8	9/9	4/7
Table 7	nlpaueb/bert-base-uncased-eurlex	Hugging Face	128_5_16	9/9	4/7
Table 8	avsolatorio/GIST-small-Embedding-v0	Hugging Face	64_3_8	9/9	4/7

The results in Table 11 show that the models have high metric values, but potentially have problems when applied to the Kaggle dataset. The models in the previous tables achieved almost perfect results, but on independently created examples, the models show an average ability to detect various attacks. In contrast to this dataset, the models trained on the Hugging Face showed much better results when it comes to detecting

malicious queries. However, although they effectively detect malicious queries and generalize well, the models become sensitive to harmless queries, and even sometimes classify them as attacks.

Generalization analysis: A multilingual evaluation. To analyze how well the models are able to generalize their knowledge to languages other than English, an additional test was conducted on a dataset in Serbian. This experiment is crucial for understanding the limitations of existing language models (if any). For the purposes of this test, the **malignant_srp.xslx** dataset will be used. It was created by machine translation (using Microsoft Bing tools) of the original Hugging Face dataset from English to Serbian. A sample of 50% of the data was randomly selected from the translated dataset. Tables 10, 11, and 12 show comparative performance results of all models when testing their generalization power on the Serbian language dataset

Table 10 Results of the nlpaueb/bert-base-uncased-eurlex model on the Serbian language dataset

Hyperparameters	Results (Hugging Face)	Results (Kaggle)
128_3_8	Accuracy: 0. 3049	Accuracy: 0. 9845
	Precision: 0.1 238	Precision: 0. 9
	Recall: 1.0	Recall: 0.9474
	AUC: 0.8913	AUC: 0.9958
256_3_8	Accuracy: 0. 4264	Accuracy: 0. 9742
	Precision: 0. 1462	Precision: 0. 811
	Recall: 1.0	Recall: 0.9605
	AUC: 0.9644	AUC: 0.9962
128_5_16	Accuracy: 0.2 610	Accuracy: 0. 9483
	Precision: 0. 1173	Precision: 0. 9737
	Recall: 1.0	Recall: 0.4868
	AUC: 0.9749	AUC: 0.9954
64_3_8	Accuracy: 0.2765	Accuracy: 0. 9587
	Precision: 0.1195	Precision: 0. 7115
	Recall: 1.0	Recall: 0.9737
	AUC: 0.9690	AUC: 0.9946

Table 11 Results of the avsolatorio/GIST-small-Embedding-v0 model on the Serbian language dataset

Hyperparameters	Results (Hugging Face)	Results (Kaggle)
128_3_8	Accuracy: 0.2842	Accuracy: 0. 98 71
	Accuracy: 0.1 2 06	Accuracy: 0.9 024
	Recall: 1.0	Recall: 0.9 737
	AUC: 0. 7752	AUC: 0.99 69
256_3_8	Accuracy: 0.2829	Accuracy: 0. 9 587
	Accuracy: 0. 1 204	Precision: 0.7075
	Recall: 1.0	Recall: 0. 9868
	AUC: 0. 7545	AUC: 0.99 78
128_5_16	Accuracy: 0.5078	Accuracy: 0. 9 638
	Precision: 0.1663	Precision: 0.7353
	Recall: 1.0	Recall: 0. 9868
	AUC: 0.9 249	AUC: 0. 9977
64_3_8	Accuracy: 0.4238	Accuracy: 0. 9 599
	Precision: 0.1456	Precision: 0.7143
	Recall: 1.0	Recall: 0. 9868
	AUC: 0. 8961	AUC: 0. 9959

Table 12 Results of the sentence-transformers/all-MiniLM-L6-v2 model on the Serbian language dataset

Hyperparameters	Results (Hugging Face)	Results (Kaggle)
128_3_8	Accuracy: 0.3411	Accuracy: 0.9587
	Precision: 0.1297	Precision: 0.7075
	Recall: 1.0	Recall: 0. 9868
	AUC: 0. 9014	AUC: 0. 9927
256_3_8	Accuracy: 0.3256	Accuracy: 0.9625
	Precision: 0.1271	Precision: 0.7282
	Recall: 1.0	Recall: 0. 9868
	AUC: 0. 9149	AUC: 0. 9944
128_5_16	Accuracy: 0.3708	Accuracy: 0.9574
	Precision: 0.1350	Precision: 0.7009
	Recall: 1.0	Recall: 0. 9868
	AUC: 0. 9334	AUC: 0. 9929
64_3_8	Accuracy: 0.3204	Accuracy: 0.9134
	Precision: 0.1262	Precision: 0.5315
	Recall: 1.0	Recall: 1.0
	AUC: 0. 9333	AUC: 0.9 878

Models trained exclusively in English are unable to perform semantic analysis or reliably distinguish benign from malicious queries in the Serbian language. Rather than understanding the actual intent of the query, the models seem to automatically treat

unfamiliar linguistic structure as an anomaly and classify it as a threat by default. This further confirms the key finding that the models did not learn a language-agnostic, abstract concept of attack, but were strictly limited to the linguistic patterns of English they saw during training.

Finally, another analysis was conducted to examine the robustness of the models and their generalization ability, but so that the models trained on Kaggle dataset tested on Hugging Face set and vice versa. Another reason for this analysis is to determine whether and to what extent the models are overtrained on the specific data distribution of their training set. The results of this cross-validation of the models are shown in Table 13 and Table 14, which show the models with the combination of hyperparameters max_length =128, epochs =3, batch_size =8 .

Table 13 Performance of models trained on Hugging Face data and tested on Kaggle data

Model	Accuracy	Precision	Recall	F1	AUC
nlpaueb/bert-base-uncased-eurlex	0.7173	0.3080	1.0	0.4710	0.9908
avsolatorio/GIST-small-Embedding-v0	0.7830	0.3672	1.0	0.5371	0.9765
sentence-transformers/all-MiniLM-L6-v2	0.6116	0.2448	1.0	0.3933	0.9567

Table 14 Performance of models trained on Kaggle data and tested on Hugging Face data

Model	Accuracy	Precision	Recall	F1	AUC
nlpaueb/bert-base-uncased-eurlex	0. 6648	1.0	0.0985	0.1794	0.7591
avsolatorio/GIST-small-Embedding-v0	0. 6648	0.9167	0.1084	0.1938	0.6234
sentence-transformers/all-MiniLM-L6-v2	0.7216	0.8148	0.3251	0.4648	0.6559

The cross-test results (detailed in Tables 13 and 14) reveal a key finding. The models demonstrate high precision, which indicates that the number of false positive alarms (false alarms) was negligible. When the model detected an attack, it was almost always right.

However, this finding is in sharp contrast to the extremely low recall. The results show that the models miss the vast majority of real attacks, in some cases close to 90%.

This confirms the conclusion reached in the test with the Serbian language: the models failed to learn and generalize the abstract concept of attack. Instead, they overfitted and learned to recognize specific linguistic patterns and keywords present exclusively in their training set, while missing all new and unprecedented attack variants.

Therefore, the results clearly indicate that no single dataset is sufficient for training. Building a robust query injection detection system requires training on significantly

more voluminous and linguistically diverse data, covering a wider range of both explicit and more subtle, paraphrased attacks.

5 Conclusion

This paper presented a modular and efficient system for proactive cybersecurity, specifically designed for the detection of prompt injection attacks with a high degree of accuracy. We have addressed a critical security vulnerability of large language models (LLMs) arising from their susceptibility to manipulative inputs. The core of our system is a multi-stage pipeline that combines advanced, fine-tuned BERT classifiers (like all-MiniLM-L6-v2 and GIST-small), pre-trained on large public datasets (Kaggle and Hugging Face), with a robust binary classification module.

The primary innovation of this research is the focus on attack detection as an external, proactive line of defense, rather than relying on the internal security of the LLM itself. Performance evaluation demonstrated a key finding: although the system achieves extremely high accuracy (often >95%) on data originating from the same domain as the training data, its main limitation is poor generalization ability.

Our research quantitatively proved a drastic drop in performance (especially low Recall of ~10-15%) when cross-testing on different datasets, as well as a complete failure on multilingual (Serbian) data. This proves that the models did not learn the abstract concept of attack, but over-adapted to specific linguistic patterns. With measured high precision (very low false positives) but critically low Recall (high false negatives), this system provides a practical and robust basis for understanding the limitations of simple classifiers and lays the groundwork for future improvements.

Future work must include training on significantly larger and more linguistically diverse datasets to improve generalization, as well as investigating the effectiveness of models specifically trained for the cybersecurity domain, such as SecBERT or CyberBERT.

Acknowledgments. The authors express their gratitude to Metropolitan University for the stimulating environment for scientific research and for the financial support provided. Particular gratitude is owed to the measure of exempting the authors from the registration fee, which directly enabled the publication and presentation of the results of this research.

Disclosure of Interests. The authors hereby confirm that no known financial or non-financial competing interests exist that could be considered a potential conflict of interest in relation to the research presented in this paper. Accordingly, the authors declare the absence of any personal, professional, or financial relationships that could unduly influence the objectivity or integrity of the presented findings and conclusions.

References

1. OWASP. (2023). OWASP Cannon 10 for LLM Applications. https://llmtop10.com/
2. Kaddour, J., Harris, J., Mozes, M., Bradley, H., Raileanu, R., and McHardy, R. (2023). Challenges and Applications of Large Language Models, 2307.10169.

3. Rahman, MA, Shahriar, H., Wu, F. Cuzzocrea, A. (2024). Applying Pre-trained Multilingual BERT in Embeddings for Improved Malicious Prompt Injection Attacks Detection. University of West Florida, Tuskegee University, University of Calabria.
4. Yu, J., Lynn, X., Yu, Z., & Xing, X. (2024). GPTFUZZER: Order Teaming large language models with auto-generated jailbreak prompts. arXiv preprint arXiv:2309.10253. https://arxiv.org/pdf/2309.10253
5. Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.
6. arXiv preprint arXiv:1810.04805
7. Deepset. (2024). deepset/prompt-injections [Dataset]. Hugging Face. https://huggingface.co/datasets/deepset/prompt-injections
8. Hugging Face. (2025). Sequence classification. https://huggingface.co/docs/transformers/tasks/sequence_classification
9. Gupta, M., (2025). What with Prompt Injection? AI has Gothic and new poison
10. A bigger problem then LLM Hallucinations. Medium. [https://medium.com/data-science-in-your-pocket/what-is-prompt-injection-ai-has-got-a-new-poison-3b6455b57b4d]
11. Vaswani, A., Shazeer, N., Parmar, N., Faster, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. arXiv preprint arXiv:1706.03762. https://arxiv.org/abs/1706.03762
12. GeeksforGeeks. (2025, July 17). Explanation of BERT Model – NLP. GeeksforGeeks. https://www.geeksforgeeks.org/nlp/explanation-of-bert-model-nlp/
13. Perez, F., Ribeiro, I. (2022). Ignore Previous Prompt: Attack Techniques For Language Models
14. OWASP Gen AI Security Project. (2025). LLM01: Prompt Injection. OWASP Gen AI Security Project. https://genai.owasp.org/llmrisk/llm01-prompt-injection/
15. Jain, N., Schwarzschild, A., Wen, Y., Somepalli, G., Kirchenbauer, J., Chiang, P., Goldblum, M., Saha, A., Geiping, J., Goldstein, A. (2023). Baseline defense for adversarial attacks against aligned language models.
16. arXiv preprint arXiv:2309.00614.
17. Hung, K.-H., Ko, C.-Y., Rawat, A., Chung, I.-H., Hsu, WH, & Chen, P.-Y. (2024). Attention Tracker: Detecting prompt injection attacks in LLMs. North American Chapter of the Association for Computational Linguistics
18. Kaggle. (n.d.). Kaggle: Your machine learning and data science community . Kaggle. https://www.kaggle.com
19. Hugging Face. (n.d.). Hugging Face - The AI community building the future. Hugging Face. https://huggingface.co
20. Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Aletras, N., & Androutsopoulos, I. (2020). {LEGAL}-{BERT}: The Muppets straight out of Law School. In *Findings of the Association for Computational Linguistics: EMNLP 2020* (Short Papers) (pp. 2898–2904). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.261
21. Sentence Transformers. (n.d.). all-MiniLM-L6-v2 [Model]. Hugging Face. https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
22. Solatorio, AV (2024). GISTEmbed: Guided In-sample Selection of Training Negatives for Text Embedding Fine-tuning [Preprint]. arXiv. https://arxiv.org/abs/2402.16829