People's Democratic Republic of Algeria

Ministery of Higher Education and Scientific Research

Ferhat Abbas University of Setif

Faculty of Sciences

Informatics Department



**DISSERTATION**

Presented in fulfillment of the requirements of obtaining the degree

**Master 2 in Informatics**

Specialty: Data Engineering and Web Technologies

**THEME**

Detecting SQL injections using BERT

*Presented by:*

ATHMANI RAMI

BOUHEZILA NASSIM

*Publicly defended on: dd/mm/yyyy*

*In front of the jury composed of:*

**Supervisor: Mr.** BENZINE Mehdi. MC UFAS

**2022/2023**

Dedication

*To our parents,*

*To our grandparents,*

*To our brothers and sisters,*

*To our entire family,*

*To all our friends.*

*Athmani Rami,  
Bouhezila Nassim.*

Abstract

Deep learning techniques have improved various domains by using their ability to learn complex patterns from large datasets. In this dissertation, we employed the power of Deep Learning, specifically BERT language model (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers), to resolve the issue of SQL injection attacks on web applications.

The goal of our study is to develop a Deep Learning model using BERT that can accurately identify SQL injections.

Based on the results, our model demonstrated excellent performance; it also indicated that BERT outperforms the compared machine learning models across different evaluation metrics. These results affirm the effectiveness of BERT in detecting SQL injection attacks, underscoring its superior performance in our study.

Résumé

Les techniques d'apprentissage profond ont amélioré divers domaines en utilisant leur capacité à apprendre des complexes patterns à partir de grands ensembles de données. Dans ce mémoire, nous avons utilisé la puissance de l'apprentissage profond, en particulier le modèle de langage BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers), pour résolu le problème des attaques par injection SQL sur les applications web.

L'objectif de notre étude est de développer un modèle d'apprentissage profond en utilisant BERT qui identifier les injections SQL avec précision.

D'après les résultats, notre modèle a démontré d'excellentes performances ; ils ont également indiqués que BERT a surpassé les autres modèles d'apprentissage automatique comparés à travers différents métriques d'évaluation. Ces résultats confirment l'efficacité de BERT dans la détection des attaques par injection SQL, confirmer sa performance supérieure dans notre étude.

**ملخص**

حسنت تقنيات التعلم العميق مجالات مختلفة باستخدام قدرتها على تعلم الأنماط المعقدة من مجموعات البيانات الكبيرة. في هذه الأطروحة، استخدمنا قوة التعلم العميق، وتحديداً نموذج اللغة "BERT" (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) ، لحل مشكلة هجمات حقن SQL على تطبيقات الويب.

الهدف من دراستنا هو تطوير نموذج التعلم العميق باستخدام BERT الذي يمكنه تحديد هجمات حقن SQL بدقة.

بناءً على النتائج المتحصل عليها، أظهر نموذجنا أداءً ممتازًا؛ كما أشار إلى أن BERT يتفوق في الأداء على نماذج التعلم الآلي التي تم المقارنَة بها عبر مقاييس التقييم المختلفة. تؤكد هذه النتائج فعالية BERT في اكتشاف هجمات حقن SQL ، مما يؤكد أدائها المتفوق في دراستنا.

Table of contents

General Introduction ……………………………………………………...10

Chapter 1 SQL Injections 12

[1.1 Introduction 12](#_Toc136903116)

[1.2 Understanding how web applications work 13](#_Toc136903117)

[1.3 How SQL injections work 14](#_Toc136903118)

[1.3.1 Definition 14](#_Toc136903119)

[1.4 Techniques of SQL injections 16](#_Toc136903120)

[1.4.1 Tautologies 16](#_Toc136903121)

[1.4.2 Error-based SQL injection 16](#_Toc136903122)

1.4.2.1 Mysql database errors 17

[1.4.3 Blind SQL injection 21](#_Toc136903123)

[1.4.3.1 Content-based 21](#_Toc137587264)

[1.4.3.2 Time-based 22](#_Toc137587265)

[1.4.4 Union-based SQL injections 23](#_Toc136903124)

[1.5 SQL injection defense techniques 25](#_Toc136903125)

[1.5.1 Escaping 25](#_Toc136903126)

[1.5.2 Input validation 25](#_Toc136903127)

[1.5.3 Parameterized queries 27](#_Toc136903128)

[1.5.4 Web Application firewalls WAF 28](#_Toc136903129)

1.5.5 Detection using machine learning 28

[1.6 Conclusion 29](#_Toc136903131)

[Chapter 2 Deep Learning 30](#_Toc136904958)

[2.1 Introduction 30](#_Toc136904959)

[2.2 Machine learning 30](#_Toc136904960)

[2.2.1 Types of machine learning…. …..31](#_Toc136904961)

[2.2.1.1 Supervised learning………………………………………………….......31](#_Toc136904962)

[2.2.1.2 Unsupervised learning…….………………………………………...…...32](#_Toc136904963)

[2.2.1.3 Reinforcement ………………………………………………………...32](#_Toc136904964)

[2.2.2 Machine learning algorithms ….32](#_Toc136904965)

[2.2.2.1 Linear regression………………………………………………………...32](#_Toc136904966)

[2.2.2.2 Logistic regression………………………………………………..……..33](#_Toc136904967)

[2.2.2.3 Support vector machines…………………………………………...……34](#_Toc136904968)

[2.2.2.4 K-Means…………………………………………………………...…….34](#_Toc136904969)

[2.2.3 Machine learning applications 35](#_Toc136904970)

[2.3 Deep learning 35](#_Toc136904971)

[2.3.1 Activation functions 36](#_Toc136904972)

[2.3.2 Deep Neural Networks 37](#_Toc136904973)

[2.3.3 Deep learning architectures 37](#_Toc136904974)

[2.3.3.1 Recurrent Neural Networks……………………………………………...38](#_Toc136904975)

[2.3.3.2 Long Short-Term Memory Networks ………………………………...38](#_Toc136904976)

[2.3.3.3 Gated Recurrent Units …………………………………………………39](#_Toc136904977)

[2.3.3.4 Transformers …………………………………………………………40](#_Toc136904978)

[2.3.4 Deep learning applications 43](#_Toc136904979)

[2.4 Conclusion 44](#_Toc136904980)

[**Chapter 3 Conception and Implimentation 45**](#_Toc136904409)

[3.1 Introduction 45](#_Toc136904410)

[3.2 General conception of the solution 45](#_Toc136904411)

[3.3 Chosen model: BERT 46](#_Toc136904412)

[3.3.1 BERT architecture 48](#_Toc136904413)

[3.3.2 BERT for text classification 48](#_Toc136904414)

[3.3.3 Why BERT was chosen 49](#_Toc136904415)

[3.3.4 Fine-tuning BERT for SQL injection detection 49](#_Toc136904416)

[3.4 Presentation of development tools 49](#_Toc136904417)

[3.4.1 Programming language 49](#_Toc136904418)

[3.4.2 Development environment 50](#_Toc136904419)

[3.5 Dataset 51](#_Toc136904420)

[3.6 Code and implementation 53](#_Toc136904421)

[3.6.1 Import necessary libraries and tools 53](#_Toc136904422)

[3.6.2 Split and preprocess data for the BERT model 54](#_Toc136904423)

[3.6.3 Build the BERT model 55](#_Toc136904424)

[3.6.4 Train the BERT model 55](#_Toc136904425)

[3.6.5 Make predictions with the BERT model 56](#_Toc136904426)

[3.7 Choice of hyperparameters 57](#_Toc136904427)

[3.7.1 Preprocessing hyperparameters 57](#_Toc136904428)

[3.7.2 Model training hyperparameters 57](#_Toc136904429)

[3.7.3 Data split hyperparameters 57](#_Toc136904430)

[3.8 Conclusion 58](#_Toc136904431)

[Chapter 4 Test and Evaluation 59](#_Toc136904741)

[4.1 Introduction 59](#_Toc136904742)

[4.2 Evaluation metrics for assessing model performance 59](#_Toc136904743)

[4.2.1 Confusion matrix 59](#_Toc136904744)

[4.2.2 Accuracy 60](#_Toc136904745)

[4.2.3 Precision 60](#_Toc136904746)

[4.2.4 Recall 61](#_Toc136904747)

[4.2.5 F1 Score 61](#_Toc136904748)

[4.3 Model performance analysis 61](#_Toc136904749)

[4.4 Comparative analysis with other approaches 63](#_Toc136904750)

[4.5 Model performance evaluation on new data 64](#_Toc136904751)

[4.6 Conclusion 64](#_Toc136904752)

General Conclusion 65

References ……………………………………………………...…………..67

**List of figures**

[Figure ‎1.1 Three tier architecture. 13](file:///D:\New-G-Drive\Master-2\Soutenance\detecting-sql-injections-using-bert\Memoire\Chapter-01-SQL-Injection.docx#_Toc136903135)

[Figure 1.2 Example of a SQL injection attack. 15](file:///D:\New-G-Drive\Master-2\Soutenance\detecting-sql-injections-using-bert\Memoire\Chapter-01-SQL-Injection.docx#_Toc136903136)

[Figure 1.3 How information flows during an SQL injection error. 16](file:///D:\New-G-Drive\Master-2\Soutenance\detecting-sql-injections-using-bert\Memoire\Chapter-01-SQL-Injection.docx#_Toc136903137)

[Figure ‎1.4 SQL injection vulnerability in PHP code. 18](file:///D:\New-G-Drive\Master-2\Soutenance\detecting-sql-injections-using-bert\Memoire\Chapter-01-SQL-Injection.docx#_Toc136903138)

[Figure ‎1.5 handle query error with mysql\_error() function in PHP. 19](file:///D:\New-G-Drive\Master-2\Soutenance\detecting-sql-injections-using-bert\Memoire\Chapter-01-SQL-Injection.docx#_Toc136903139)

[Figure ‎1.6 Character-escaping in PHP code example. 25](file:///D:\New-G-Drive\Master-2\Soutenance\detecting-sql-injections-using-bert\Memoire\Chapter-01-SQL-Injection.docx#_Toc136903140)

[Figure ‎1.7 Parameterized queries using PDO in PHP code example. 27](file:///D:\New-G-Drive\Master-2\Soutenance\detecting-sql-injections-using-bert\Memoire\Chapter-01-SQL-Injection.docx#_Toc136903141)

[Figure ‎2.1 Graphical representation of linear regression. 33](file:///D:\New-G-Drive\Master-2\Soutenance\detecting-sql-injections-using-bert\Memoire\Chapter-02-Deep-Learning.docx#_Toc136903515)

[Figure ‎2.2 Graphical representation of logistic regression. 33](file:///D:\New-G-Drive\Master-2\Soutenance\detecting-sql-injections-using-bert\Memoire\Chapter-02-Deep-Learning.docx#_Toc136903516)

[Figure ‎2.3 Graphical representation of support vector machines. 34](file:///D:\New-G-Drive\Master-2\Soutenance\detecting-sql-injections-using-bert\Memoire\Chapter-02-Deep-Learning.docx#_Toc136903517)

[Figure ‎2.4 Graphical representation of k means. 35](file:///D:\New-G-Drive\Master-2\Soutenance\detecting-sql-injections-using-bert\Memoire\Chapter-02-Deep-Learning.docx#_Toc136903518)

[Figure ‎2.5 Schematic representation of a neural network. 37](file:///D:\New-G-Drive\Master-2\Soutenance\detecting-sql-injections-using-bert\Memoire\Chapter-02-Deep-Learning.docx#_Toc136903519)

[Figure ‎2.6 Architecture of transformers. 40](file:///D:\New-G-Drive\Master-2\Soutenance\detecting-sql-injections-using-bert\Memoire\Chapter-02-Deep-Learning.docx#_Toc136903520)

[Figure ‎3.1 Sql injection detection tool conception and architecture. 46](file:///C:\Users\dell\Downloads\Chapter-03-conception-implementation.docx#_Toc136904628)

[Figure ‎3.2 BERT model size. 47](file:///C:\Users\dell\Downloads\Chapter-03-conception-implementation.docx#_Toc136904629)

[Figure ‎3.3 BERT model architecture. 48](file:///C:\Users\dell\Downloads\Chapter-03-conception-implementation.docx#_Toc136904630)

[Figure ‎3.4 Dataset query classes distribution…..………………………………………….52](file:///D:\New-G-Drive\Master-2\Soutenance\detecting-sql-injections-using-bert\Memoire\Chapter-03-conception-implementation.docx#_Toc136728390)

[Figure ‎3.5 Split data into training and testing set with BERT data preprocess Python code. 54](file:///D:\New-G-Drive\Master-2\Soutenance\detecting-sql-injections-using-bert\Memoire\Chapter-03-conception-implementation.docx#_Toc136728391)

[Figure ‎3.6 Build BERT model Python code. 55](file:///D:\New-G-Drive\Master-2\Soutenance\detecting-sql-injections-using-bert\Memoire\Chapter-03-conception-implementation.docx#_Toc136728392)

[Figure ‎3.7 Train BERT model Python code. 55](file:///D:\New-G-Drive\Master-2\Soutenance\detecting-sql-injections-using-bert\Memoire\Chapter-03-conception-implementation.docx#_Toc136728392)

[Figure ‎3.8 Make predictions with the trained model. 56](file:///D:\New-G-Drive\Master-2\Soutenance\detecting-sql-injections-using-bert\Memoire\Chapter-03-conception-implementation.docx#_Toc136728393)

**General Introduction**

The rapid growth of web applications has revolutionized the way we interact and conduct various activities online. From e-commerce platforms and social networks to financial systems and government portals, web applications have become an integral part of our daily lives. However, with greater dependence on online applications comes an increased danger of cyber attacks with SQL injections being one of the most common and dangerous vulnerabilities.

To mitigate the growing threat of SQL injections, traditional approaches such as input validation and query parameterization have been widely adopted. While these methods provide some level of protection, they often struggle to keep pace with the evolving attack techniques employed by adversaries. Thus, there is a pressing need for more advanced and proactive defense mechanisms to detect and prevent SQL injection attacks.

In recent years, deep learning approaches have emerged as a promising solution in various domains, leveraging their ability to automatically learn complex patterns from large datasets. One such powerful deep learning model is BERT (Bidirectional Encoder Representations from Transformers), originally developed for natural language processing tasks. BERT has proven to be highly effective in capturing the semantic and contextual understanding of text, leading to remarkable performance in tasks such as text classification.

In this research, we propose using the power of BERT-based deep learning models to address the critical issue of SQL injection attacks. Our objective is to develop a reliable and efficient detection model capable of accurately identifying SQL injection attempts in real-time. By using BERT's contextual understanding and semantic representation capabilities, we aim to create a model that can effectively distinguish between normal and SQL malicious queries.

Our thesis is organized as follows:

In Chapter 1, we exploreSQL injection attacks, their definitions, types and their detecting techniques, then we head on machine learning and Deep Learning, we present popular algorithmic approaches in machine learning and explore deep learning architectures in Chapter 2. Chapter 3 is dedicated to the general conception of our work and the materials used, including the dataset and the type of deep learning architecture employed. Furthermore, we cover the preprocessing steps taken to ensure the accuracy and efficiency of our system. In Chapter 4, we discuss the test and evaluation of our model for detecting SQL injection attacks. We use a variety of evaluation metrics, including accuracy, precision, recall, and F1 score. We also compare the performance of our model to other machine learning algorithms and related works.