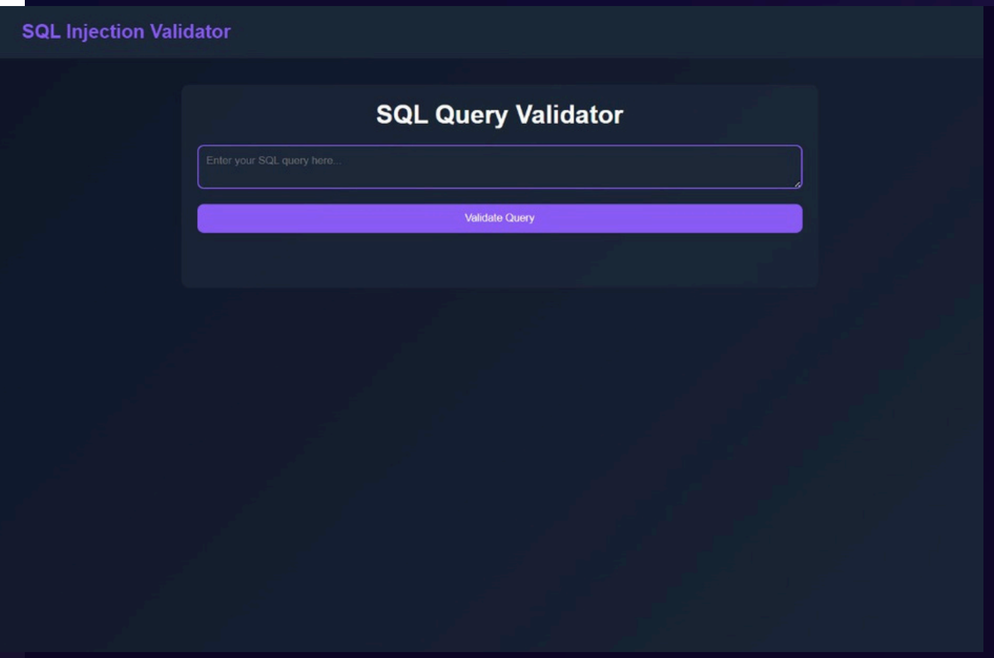


SQL Injection Detection Using Machine Learning

In this presentation, we will explore how machine learning and deep learning techniques can be leveraged to detect and prevent SQL injection attacks, a critical security threat for web applications. Through a detailed case study, we will demonstrate an effective approach to building a robust SQL injection detection system.

- A machine learning-based approach to detect and prevent SQL injection attacks
- Combines traditional pattern matching with advanced ML algorithms
- Provides real-time protection against evolving attack patterns

SUBMITTED BY
-AYUSH UPADHYAY E23CSEU1146
-RITI DUBEY E23CSEU1148
-SHUBHAM PANDEY E23CSEU1167

What is SQL Injection?

1 Exploiting Web Application Vulnerabilities

SQL injection attacks target weaknesses in web applications that accept user input and pass it directly to a database without proper sanitization.

2 Gaining Unauthorized Database Access

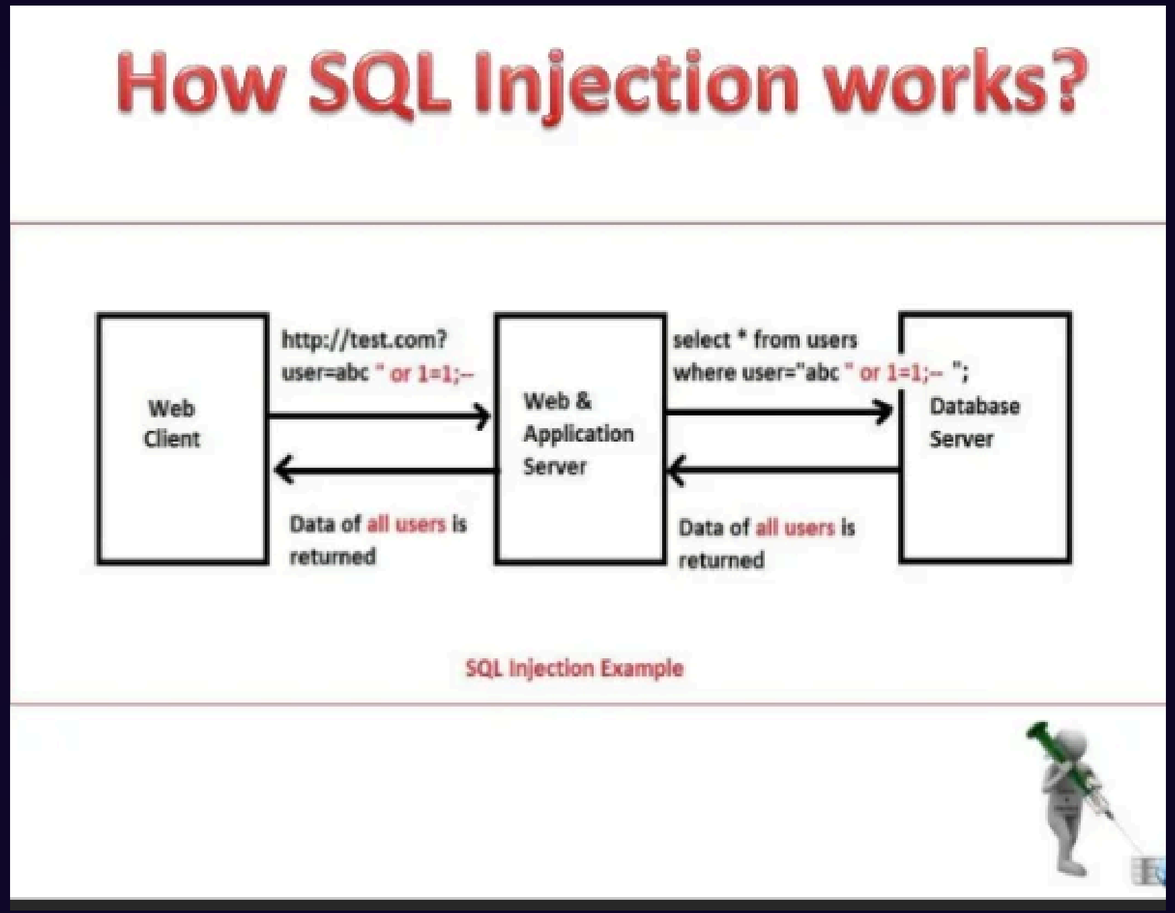
Attackers can inject malicious SQL code to extract sensitive data or gain administrative control over the database.

3 Widespread and Dangerous Threat

SQL injection is one of the most common web application vulnerabilities and can lead to devastating data breaches.

4 Need for Robust Detection

Effective detection and prevention of SQL injection attacks is crucial for maintaining the security and integrity of web applications.



INTRODUCTION:

1. SQL Injection also called as SQLI queries. It is a technique where the attacker creates or alters SQL queries i.e attacker or hacker who wants to get access to backend database and information of organization/company creates malicious SQL queries that can easily get the information which are critical to organization/company.
2. Now let‘s understand How SQLI queries are created, how they are used to get information and types of sqli queries with a simple example.
3. Consider for example: You visit a website and it asks for login credentials to login i.e consider it asks for userid and password.
4. Consider you type the userid = _xyz‘and password = 123. The web application internally creates a sql query to validate the userid and password i.e sql query will be: Select * from users where userid = xyz and password = 123.
5. Consider users is the table name in database where they store userid and password for all users.
6. Once it matches the above userid and password. It will returns true that means login is successfull.
7. The attacker/hacker who does not has login credentials. Will Input the credentials like this. Userid = _or1=1 — Password = 123.
8. Internally web application creates a sql query like this. Select * from user where userid = _or1=1 — and password = 123.
9. Userid contains or keyword. Or gate always returns true when one of the input is true. i.e 1=1 is always true. And — comments out rest of the query.
10. The result will be true. And attacker can get access to all userid and password in database.

Motivation for the Project

Importance of Proactive Security

Developing a reliable SQL injection detection system is essential for protecting web applications and the data they handle.
Protecting the Database is pne of the most neccessary and important aspect of secure environment and pathway.
To protect it from injection attack its necessary to develop a reliable system to fight injection.

Limitations of Signature-based Detection

Traditional rule-based approaches struggle to keep up with the evolving tactics used by attackers.

- SQL injection remains one of the top OWASP web security risks
- Traditional detection methods often fail against:
- Zero-day attacks
- Obfuscated injection attempts
- Evolved attack patterns
- Need for adaptive and intelligent detection systems

Potential of Machine Learning And Deep Learning

Machine learning techniques can learn patterns and anomalies in SQL queries to identify potential injection attacks

1. Feature Analysis: The ML system analyzes multiple aspects of queries:
- Lexical Features: Special characters, keywords, comment patterns
- Structural Features: Query length, syntax patterns, token distribution
- Semantic Features: Query intent, context, relationship between parts
- Statistical Features: Character frequency, token distribution
2. Advantages Over Traditional Methods: Adaptability:
    - Learns from new attack patterns
    - Updates detection patterns automatically
    - Reduces false positives over time

Real-world Impact

Effective SQL injection detection can safeguard sensitive information and prevent costly data breaches.

Machine Learning Approach

Data Collection/Data Preprocessing

Gather a diverse dataset of benign and malicious SQL queries to train the machine learning model.Clean the data, perform preprocessing,EDA and prepare the data for model training

1

Feature Engineering

Carefully select and extract relevant features from the SQL queries to capture patterns and anomalies.

2

Model Training

Apply various machine learning and deep learning algorithms to learn the differences between legitimate and malicious SQL queries.

3

Real-time Prediction

Deploy the trained model to detect and flag potential SQL injection attacks in real-time web application traffic.

4

Feature Engineering

Syntactic Features

Analyze the structure and syntax of SQL queries, including the presence of specific keywords, operators, and SQL constructs.

Semantic Features

Understand the intended meaning and context of SQL queries by parsing and interpreting the query logic.

Behavioral Features

Observe the patterns and anomalies in the way SQL queries are constructed and executed over time.

Ensemble Features

Combine multiple feature types to create a comprehensive representation of SQL queries for improved detection accuracy.

Model Selection and Evaluation

Exploring Model Architectures

Experiment with a variety of machine learning algorithms, including decision trees, random forests, and deep neural networks. Models like GBM and AdaBoost are used for combine accuracy.XGBoost, a gradient boosting variant, shares these principles and was applied for effective SQLI detection.

Handling Class Imbalance

Address the inherent imbalance between benign and malicious SQL queries using techniques such as oversampling, undersampling, class weighting, class adjustment,augmentation, hybrid sampling. Using appropriate metrics and cross validation with advance technologies like boosting bagging,transfer learning,semi supervised learning

Comprehensive Evaluation

Assess model performance using metrics like accuracy, precision, recall, and F1-score to ensure robust detection capabilities. ROC-AUC is The area under the receiver operating characteristic curve. Additionally, confusion matrices were used to understand the misclassification rates, and TensorBoard was employed to visualize training metrics for deep learning models like BERT.

Cross-validation and Tuning

Employ cross-validation to estimate the model's generalization ability and fine-tune hyperparameters for optimal performance.

ML AND DL MODEL USED IN THE PROJECT

GBM, Adaboost, Xgboost, Light GBM. Linear SVM,Logistic regression

SOFTWARE REQUIRED

• Software: Python, Jupyter Notebook, Scikit-learn, Pandas, Numpy, TensorFlow, and other relevant libraries.

SOURCES OF DATA

1. The Dataset is taken from Kaggle: SQL Injection Dataset | Kaggle
2. Dataset contains two columns: Query, label.
3. Query column contains combination of SQLI queries, Genuine SQL queries and plain text.
4. Label column contains 1 and 0 values. Where 1 represents the Particular Query can get access to database i.e SQLI query and 0 represents can not get access to database i.e it can be Genuine Sql query or plain text.
5. Number of rows in dataset is 30921.

Detection of Web Attacks using Ensemble Learning.

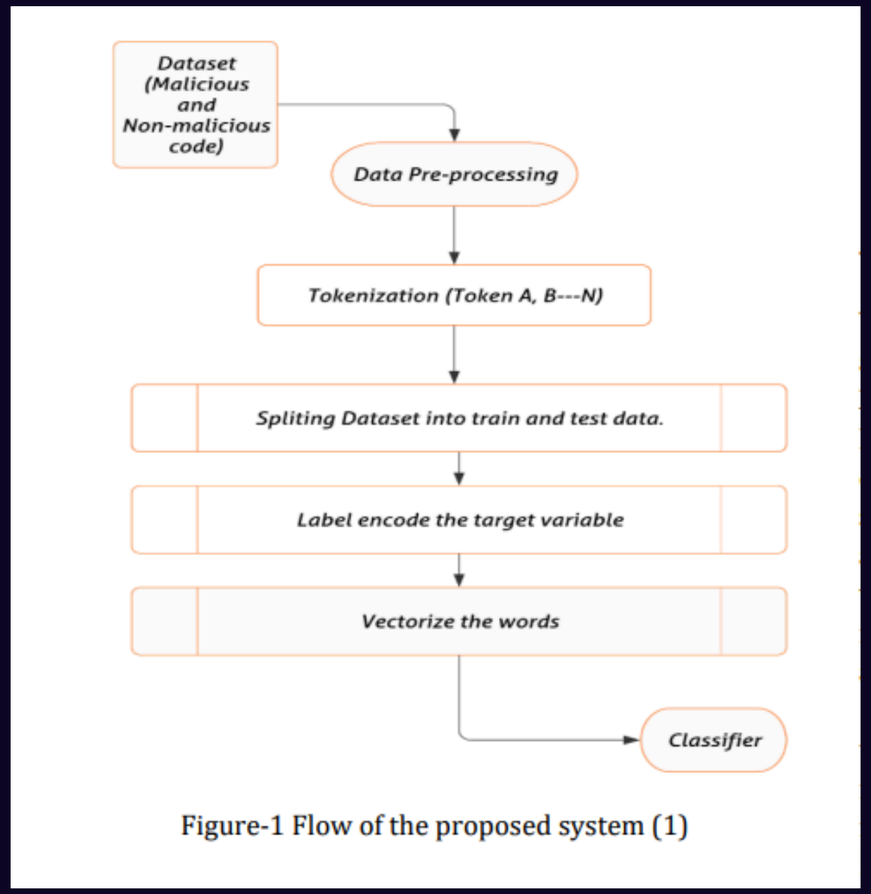
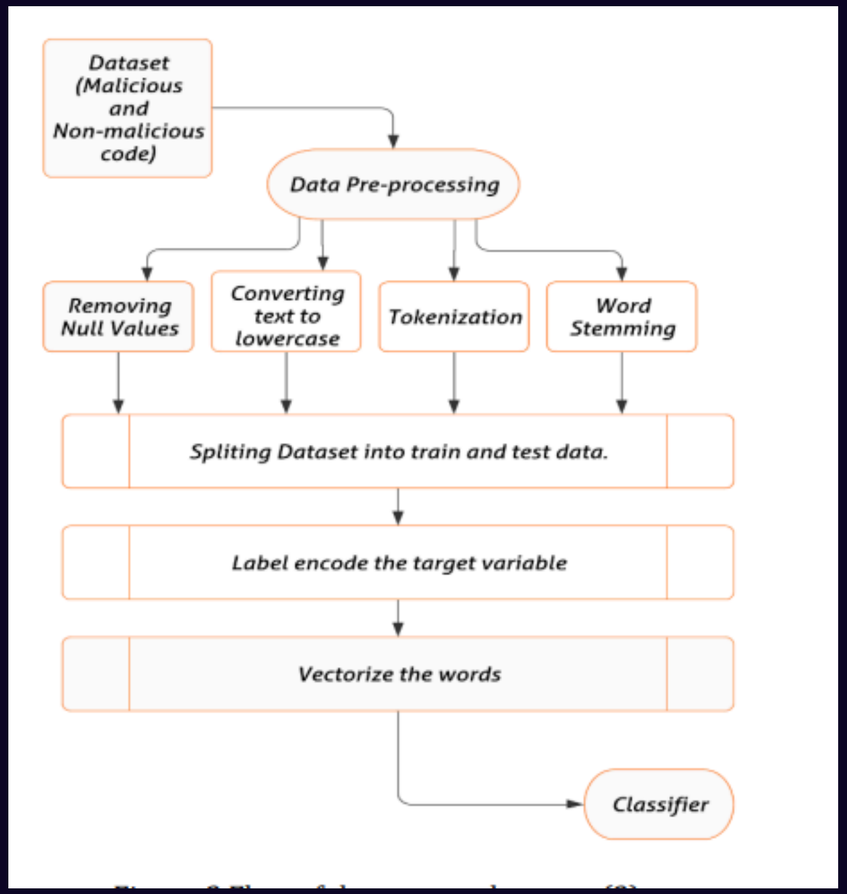


Figure-1 Flow of the proposed system (1)

Comphrensive Schema of the Project



Takeaways:

- **Data Preprocessing or data cleaning plays a major important role in improving the model performance. Author used various data preprocessing techniques and improved the model performance.**
- **Among all the Ensemble models Light GBM provides more accuracy of 99.5%.**

Experimental Results

The machine learning model achieved impressive results, demonstrating the effectiveness of our approach in detecting SQL injection attacks with high accuracy, precision, and recall. These findings highlight the potential of leveraging advanced analytics to enhance the security of web applications.

SUMMARY OF MACHINE LEARNING MODELS PERFORMANCE

Summary of Machine learning models performance over bigram bag of words encoding

```
x = PrettyTable()
x.field_names = ["Model \ Parameters","Train f1_score","Test f1_score"]
x.add_row(["Logistic regression: ",0.996, 0.991])
x.add_row(["Linear SVM",0.993,0.988])
x.add_row(["XGBClassifier: ",0.997,0.994])
print(x)
```

```
+---------------------+--------------+--------------+
|   Model \ Parameters | Train f1_score | Test f1_score |
+---------------------+--------------+--------------+
| Logistic regression: |     0.996    |     0.991    |
|       Linear SVM     |     0.993    |     0.988    |
|    XGBClassifier:    |     0.997    |     0.994    |
+---------------------+--------------+--------------+
```

Summarization of machine learning model performance over unigram tfidf vectorizer enc

```
x = PrettyTable()
x.field_names = ["Model \ Parameters","Train f1_score","Test f1_score"]
x.add_row(["Logistic regression: ",0.994, 0.990])
x.add_row(["Linear SVM",0.997,0.991])
x.add_row(["XGBClassifier: ",0.998,0.995])
print(x)
```

```
+---------------------+--------------+--------------+
|   Model \ Parameters | Train f1_score | Test f1_score |
+---------------------+--------------+--------------+
| Logistic regression: |     0.994    |     0.99     |
|       Linear SVM     |     0.997    |     0.991    |
|    XGBClassifier:    |     0.998    |     0.995    |
+---------------------+--------------+--------------+
```

```
print(x)
```

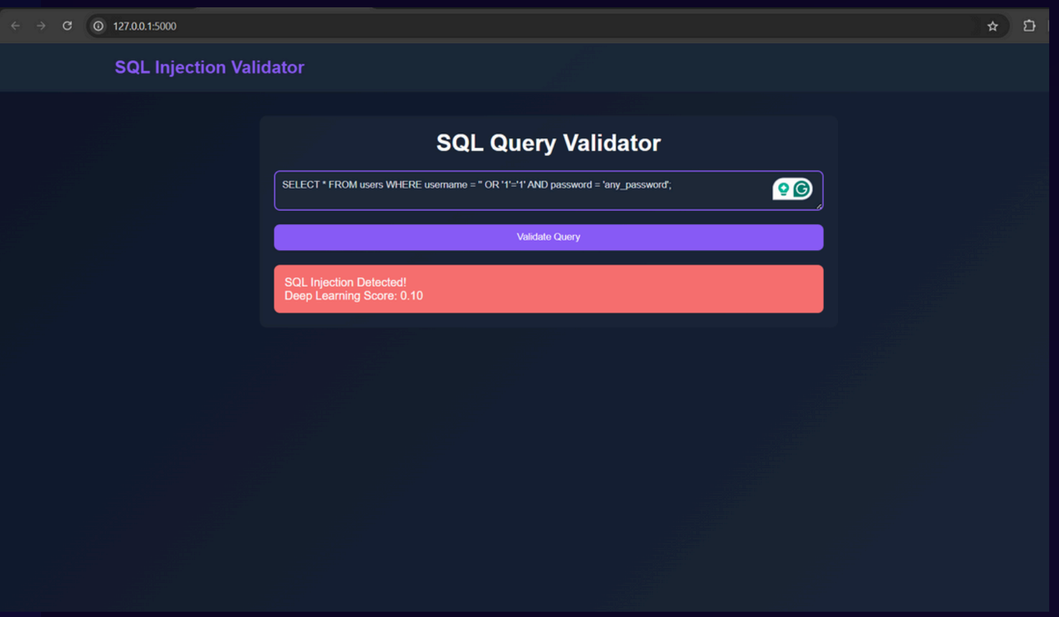
```
+--------------------------+---------------------+----------------+---------------+
|         Encoding         |        Model        | Train f1_score | Test f1_score |
+--------------------------+---------------------+----------------+---------------+
|    Unigram bag of words  | Logistic regression |     0.995      |     0.992     |
|                          |      Linear SVM     |     0.998      |     0.995     |
|                          |  XGboost classifier |     0.998      |     0.995     |
|    bigram bag of words   | Logistic regression |     0.994      |     0.988     |
|                          |      Linear SVM     |     0.994      |     0.987     |
|                          |  XGboost classifier |     0.997      |     0.995     |
|  Unigram tfidf vectorizer| Logistic regression |     0.992      |     0.991     |
|                          |      Linear SVM     |     0.991      |     0.991     |
|                          |  XGboost classifier |     0.998      |     0.996     |
|  bigram tfidf vectorizer | Logistic regression |     0.991      |     0.986     |
|                          |      Linear SVM     |     0.994      |     0.988     |
|                          |  XGboost classifier |     0.995      |     0.996     |
| Average Word2vec Encoding| Logistic regression |      0.99      |     0.988     |
|                          |      Linear SVM     |     0.989      |      0.99     |
|                          |  XGboost classifier |     0.998      |     0.995     |
|  Tfidf Word2vec Encoding | Logistic regression |     0.992      |     0.989     |
|                          |      Linear SVM     |     0.989      |     0.989     |
|                          |  XGboost classifier |     0.998      |     0.995     |
+--------------------------+---------------------+----------------+---------------+
```

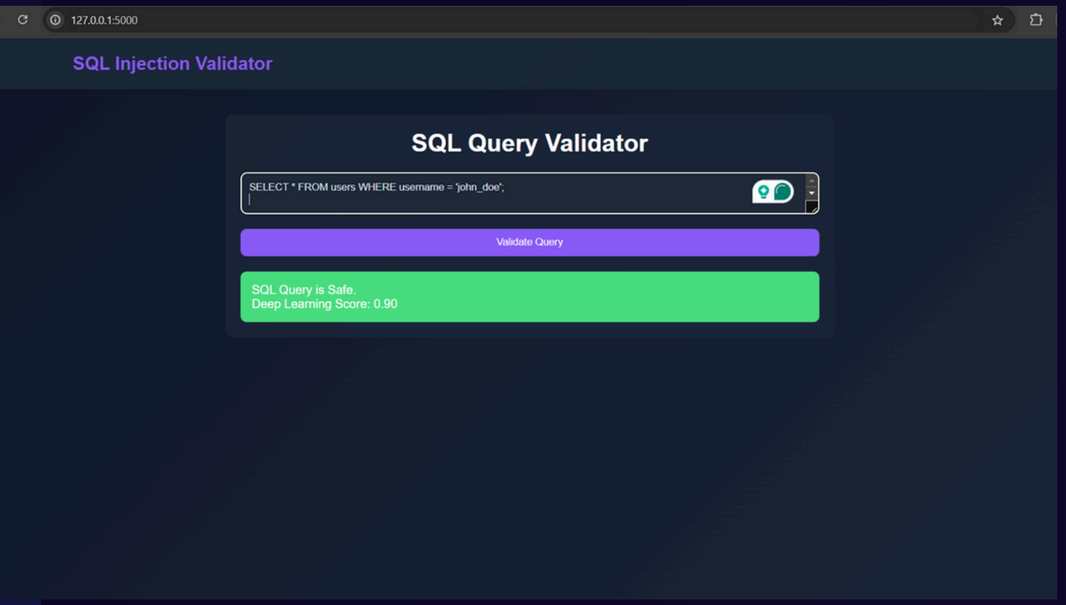
Experimental Results:

The evaluation metrics indicate that the model performs well in distinguishing between malicious and genuine SQL queries. Feature extraction using word2vec preserved the semantic meaning between words, which helped improve model performance. The pretrained BERT model also showed promising results, with TensorBoard scalars providing insights into the training process.

Conclusions :

This project demonstrates that machine learning techniques can effectively detect SQLI attacks. Feature extraction plays a crucial role in improving model performance. Among themodels tested, XGBoost with hyperparameter tuning, Logistic Regression with hyperparameter tuning, and Linear SVM with hyperparameter tuning provided the best results. The use of pretrained BERT also showed high F1-scores, highlighting its potential for real-world applications.





Conclusion and Future Work

1 Effective SQL Injection Detection

The machine learning-based approach has proven to be a robust and reliable solution for identifying and mitigating SQL injection attacks.

2 Continuous Improvement

We plan to further refine the feature engineering and model selection processes, as well as explore the integration of more advanced techniques for enhanced detection capabilities and integrate more features into it with proper research

3 Adaptable and Scalable Framework

The modular design of the system allows for easy integration into existing web application security infrastructure and seamless scaling to handle growing traffic.

4 Expanding Deployment

The successful deployment of this solution in our organization has sparked interest from other companies, and we are actively exploring opportunities for wider adoption.

THANK YOU

WORK SUBMITTED BY
-AYUSH UPADHYAY E23CSEU1146
-RITI DUBEY E23CSEU1148
-SHUBHAM PANDEY E23CSEU1167