Detection of SQL Injection Attacks Using Deep Learning and Machine Learning Hybrid Models

Ayah Khaldi
dept. of Computer Science
JUST University
Irbid, Jordan
amkhaldi23@cit.just.edu.jo

Amani Krieshan
dept. of Computer Science
JUST University
Irbid, Jordan
aakrieshan23@cit.just.edu.jo

Dr. Omar Almousa
dept. of Computer Science
JUST University
Irbid, Jordan
osalmousa@just.edu.jo

I. Abstract

An effective threat to all online applications is SQL injection (SQLi), which allows attackers to gain access to databases and extract sensitive data. Advanced attacks are often difficult to detect using traditional methods. This research introduces a hybrid model that combines machine learning (ML) and deep learning (DL) approaches to provide a new approach to detecting SQL injection. For models, the machine learning model utilizes Random Forest, Decision Trees, Support Vector Machines (SVM), and Logistic Regression. The DL model utilizes BERT and Roberta for feature extraction. Our hybrid model demonstrates superior performance in terms of detection accuracy compared to traditional techniques, as supported by experimental data. This study illustrates the effectiveness of combining ML and DL models to enhance online application security against SQLi attacks.

A. Keywords

SQL Injection, Deep Learning, Machine Learning, Hybrid Models, Cybersecurity, Web Application Security, Support Vector Machines, Random Forest, Decision Trees, Logistic Regression, RoBERTa, BERT.

II. Introduction

The fast growth of the Internet of Things (IoT) has resulted in several security issues that call for creative solutions. Although IoT applications—especially web-based ones—are useful and offer real-time monitoring, there are serious security risks associated with them [1]. Virtual Local Area Networks (VLANs), which split the network into smaller portions to reduce attack risks and potential breaches, are one suggested solution to these problems. This improves security [2].

For Internet of Things applications, SQL injection (SQLi) presents a serious risk. Using this method, attackers can find holes in web applications and run malicious SQL queries to steal private information [3] [4].

Artificial Intelligence (AI) has become a potent tool for bolstering IoT security in the face of these attacks. Massive data collections may be analysed by AI systems, which can then swiftly spot unusual trends and potential threats. In particular, machine learning models help Intrusion Detection Systems (IDS) by improving accuracy and efficiency through threat adaptation [5] [6].

Recent research efforts have focused on both identifying and mitigating various security vulnerabilities inherent in IoT systems. For instance, proposals for static analysis frameworks have been made to address vulnerabilities like SQL Injection, Code Injection, and weaknesses in cryptographic algorithms [1]. Additionally, studies have examined IoT protocols to uncover flaws that could be exploited by attackers for effective DDoS assaults [7]. Intrusion Detection systems have also been explored to bolster IoT security [8].

Moreover, advancements in technology offer promising avenues for enhancing the security of IoT networks. Technologies such as software-defined networking and network function virtualization are being leveraged to empower IoT systems to autonomously make security decisions [9]. Furthermore, solutions like Blockchain, machine learning, fog computing, and edge computing have emerged as potential means to fortify IoT security.

Machine learning, particularly predictive analytics, has shown promise in detecting and preventing attacks on IoT systems, including SQL Injection attacks [10]. Models such as naïve Bayes, Random Forest, decision trees, and logistic regression have been utilized to enhance IoT security [11].

Currently, Deep Learning (DL) is increasingly employed in detecting vulnerabilities due to its capacity to automatically extract and understand intricate attack features, eliminating the necessity for manual feature engineering. Nonetheless, implementing DL on IoT devices with constrained resources presents a challenge [12], as it demands substantial power and storage capabilities.

This document explores SQL injection detection based on feature extraction using deep learning models. In Section 3, there is a literature review and related work to identify the gap. Section 4 details the methodology, including data collection, cleaning, and analysis. Section 5 presents experimental results of our framework, interprets the findings, and discusses implications. Section 6 summarizes conclusions and suggests applications. This structured approach ensures clarity and coherence in presenting the study's methodology, findings, and implications.

III. Literature and Related Works

We've seen significant progress in thwarting SQL injection attacks in recent years. Although input validation and Web Application Firewalls (WAFs) have long been the standard, modern attacks have rendered these defences less effective. In response, novel techniques have surfaced [13] have surfaced, such as Ghafarian's method [14], which integrates many defensive layers for enhanced SQL injection attack detection and prevention. Furthermore, machine learning is creating new possibilities. Techniques like deep belief networks (DBNs) and deep autoencoders (AEs) have shown promise in identifying intricate attack patterns in Internet of Things (IoT) systems [15].

Intermittently, industrial IoT devices and web apps are major concerns about cybersecurity. There is a serious issue with threats like SQL injection attacks (SQLIA). The ever-evolving nature of online traffic and attack techniques is making it difficult for traditional security measures to stay up-to-date. The use of machine learning techniques is growing among researchers as a result. According to research conducted in 2017 and 2019 [10] [9], machine learning is capable of effectively identifying and preventing SQLIA through the analysis of large datasets and internet traffic patterns. Similar to this, IIoT networks are fortifying their defences against online threats by utilising big data analytics and machine learning. In this study, vulnerability analysis, machine learning approaches to threat mitigation, and intrusion detection system evaluation are all covered in detail.Notably, experiments and case studies conducted in real-world scenarios demonstrate the efficacy of machine learning-based anomaly detection systems in promptly identifying and thwarting assaults, hence opening the door for more flexible online security protocols.

The goal of our research is to provide a novel defence against SQL injection attacks in the context of the Internet of Things (IoT). Our goal is to integrate deep learning and machine learning to create a robust defensive system. Unlike earlier strategies, which were broad and addressed all internet threats equally, our focus is on preventing SQL injection assaults. The combination of deep learning and machine learning allows us to customise our method to better address the special difficulties posed by SQL injection in Internet of Things applications. Our ultimate objective is to improve IoT security by reducing the susceptibility of these devices to attackers and fostering a more secure online community for all users.

IV. Methodology

Our methodology hinges on constructing a hybrid approach that combines the capabilities of pre-trained transformer models like RoBERTa and BERT for deep learning (DL) with traditional machine learning (ML) algorithms such as support vector machines (SVM), random forests (RF), decision trees (DT), and logistic regression. Initially, we gather a diverse dataset comprising benign and malicious SQL queries. Preprocessing involves standardization and tokenization of the text data.

Subsequently, employing RoBERTa and BERT models, we extract contextualized features from the SQL queries, allowing for comprehensive pattern recognition and representation learning. These features are then utilized as input for the ML classifiers, where SVM, RF, DT, and logistic regression algorithms are trained to differentiate between benign and malicious queries.

Through ensemble learning techniques, we amalgamate the predictions of the DL and ML models to bolster the overall detection accuracy. Rigorous evaluation and validation procedures ensure the effectiveness and generalization of the hybrid approach across various datasets. This integrated methodology not only automates feature extraction with advanced DL models but also harnesses the interpretability and simplicity of traditional ML algorithms, culminating in a robust and versatile SQL injection detection system.

A. Pipeline

For our research, we curated multiple datasets specifically focused on SQL injection attacks. These datasets underwent a rigorous preprocessing phase, where we partitioned them into training (60%), testing (20%), and validation (20%) subsets. Subsequently, the training data was further segmented into four equal parts. Following this data preparation step, we meticulously assessed the dataset's balance to ensure an equitable representation of binary labels for SQL injection.

Employing deep learning (DL) models(RoBERTa and BERT), we conducted feature extraction on each of the four training subsets. Subsequently, we trained the extracted features using various machine learning (ML) models (SVM, RF, DT, and logistic regression).

Adding ensemble voting strategy that used to maximize predictive performance. With the models voting in unison to determine the final classification, any ties were broken by using the model with the greatest confidence score. This all-inclusive process, which includes everything from painstaking data curation to group voting, provides a strong foundation for automated quality control and consistent classification detection outcomes.

B. Dataset

As part of our ongoing efforts to enhance SQL injection detection in IoT environments, we are actively sourcing diverse and valuable SQL injection datasets from Kaggle. These datasets serve as crucial resources for training and evaluating machine learning models aimed at safeguarding IoT systems against malicious SQL injection attacks.

1) Data Description

We have curated a comprehensive dataset comprising approximately six SQL injection datasets sourced from Kaggle [1-6]. This collection includes datasets contributed by Syed Saqlain Hussain [16], Abu Syeed Sajid Ahmed [17], WenYa Yu [18], Mahdi Ghaemi [19], Kholood Salah [20], and Alex Trinity [21]. By combining these diverse datasets, we have amassed a substantial volume of records, encompassing a range of benign and malicious SQL queries.

The dataset consists of 419,481 rows and 2 main columns. Each row contains a single SQL query or sentence in the ”Query” column, represented as string text, and a corresponding label in the ”Label” column. The ”Label” column denotes whether the query is classified as a SQL injection attack (1) or a benign query (0).

This amalgamation of datasets serves as a valuable resource for training and evaluating machine learning models aimed at detecting SQL injection vulnerabilities in web applications.

2) Data preprocessing

In our data preprocessing phase, we conducted several essential steps to ensure the quality and integrity of our dataset before splitting it into testing and training sets.

Duplicate Removal: Initially, we identified and removed duplicate values from the dataset. This step was crucial in ensuring that each data point was unique and preventing any bias in our analysis.

Handling Missing Values: Following the removal of incomplete records or labels containing NaN values, we addressed any remaining missing values. Additionally, we detected and removed unnecessary columns with unnamed names and NaN values from some datasets to maintain data consistency.

As a result of these preprocessing steps, the dataset was refined to contain 163,437 rows and 2 columns. This streamlined dataset is now ready for further analysis and model training, free from redundancy and data integrity issues.

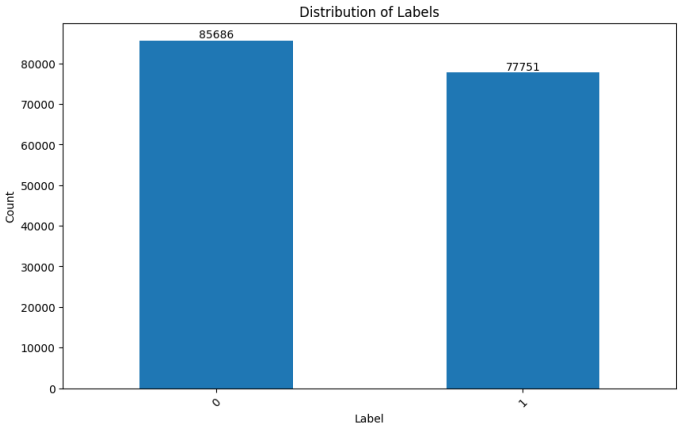

Fig. 1. Details of the labeled dataset after cleaning: The counts indicate instances of non-SQL injection labeled as 0 and instances of SQL injection labeled as 1.

In the final stage, we partitioned our dataset randomly into training, validation, and testing sets, following a split ratio of 6:2:2, as depicted in the accompanying figure, then the training dataset was divided to four datasets to .

Our cleaned dataset is available on Kaggle; this dataset url: https://www.kaggle.com/datasets/ayahkhaldi/sql-injection-dataset

C. Hybrid SQL Injection Detection Model

Hybrid SQL Injection Detection Model architecture presents a hybrid approach integrating deep learning and machine

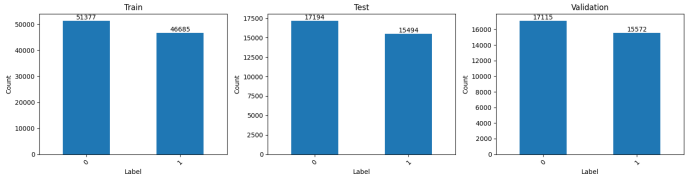

Fig. 2. Details of the labeled training, testing, and validation datasets: The counts indicate instances of non-SQL injection labeled as 0 and instances of SQL injection labeled as 1.

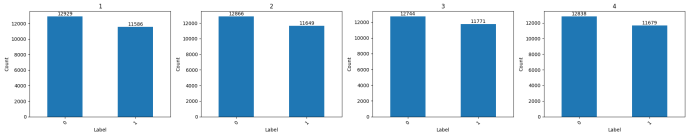

Fig. 3. Details of the labeled training datasets: The counts indicate instances of non-SQL injection labeled as 0 and instances of SQL injection labeled as 1.

learning methodologies for the detection of SQL injection attacks. Initially, the training dataset is partitioned into four segments, and feature extraction is performed using state-of-the-art deep learning models such as BERT and RoBERTa. These models are fine-tuned on SQL injection datasets to capture nuanced patterns from the text data. Subsequently, the extracted features are inputted into various machine learning classifiers including SVM, RF, DT, and logistic regression, each trained to discern between benign and malicious SQL queries. To ensure robustness, a voting mechanism aggregates the predictions of all classifiers, with the final outcome determined by majority vote. Rigorous evaluation and validation procedures validate the model’s efficacy across diverse datasets, establishing its capability to accurately detect SQL injection attacks in real-world scenarios. This hybrid architecture harnesses the complementary strengths of deep learning and machine learning, offering a reliable and versatile solution for bolstering the security of web applications against SQL injection vulnerabilities.

1) Machine Learning models

IV-C1.1 Logistic Regression

A basic machine learning approach called logistic regression has multiple benefits depending on the analytical setting. Its simplicity and interpretability are two noteworthy advantages that make it a desirable option for practitioners and researchers alike. In contrast to more intricate models, logistic regression offers clear-cut explanations of the connections between input factors and binary results. This attribute proves invaluable in explanatory tasks where understanding the underlying mechanisms is crucial [22]. Furthermore, logistic regression works incredibly well with datasets of modest to large size, providing reliable results with a low chance of overfitting [23]. Because of this feature, logistic regression is especially well-suited for scenarios in which there is a lack of available data or where interpretability and transparency of the model are crucial.

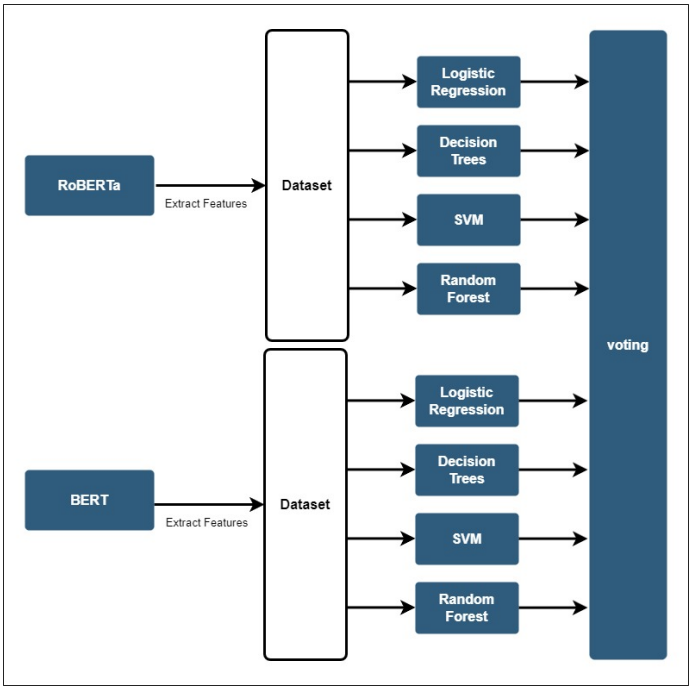

Fig. 4. Hybrid SQL Injection Detection Model: Leveraging Deep Learning for Feature Extraction and Machine Learning for Detection of SQL Injection Attack.

IV-C1.2 SVM

Strong classifiers that are frequently employed in a variety of machine learning applications are support vector machines (SVMs). SVM's ability to handle complex decision boundaries and high-dimensional data is one of its main advantages. SVM can thus be used for jobs where linear models would not be adequate, such as those involving feature-rich datasets. Furthermore, SVM performs well even with small to medium-sized datasets since it can manage situations in which there are more characteristics than samples. SVM's resistance to overfitting and good generalization performance are largely attributed to its capacity to identify the ideal hyperplane that maximizes the margin between classes, especially when data is scarce.

IV-C1.3 Decision Trees

Decision trees are popular machine learning models known for their simplicity, interpretability, and versatility. They offer an intuitive representation of decision-making processes, making them accessible to both technical and non-technical stakeholders. With minimal preprocessing requirements, decision trees can handle various types of data and capture complex relationships between features and target variables without strict assumptions about data distribution. One of the key advantages of decision trees is their ability to automatically handle feature interactions and selection, reducing the need for extensive feature engineering [24]. By recursively splitting the feature space, decision trees identify the most informative features for prediction, simplifying the modeling process and enhancing robustness against irrelevant features. Moreover, decision trees effectively capture complex decision boundaries and are less sensitive to outliers, making them suitable for diverse datasets and real-world applications.

IV-C1.4 Random Forest

Because of its strength and versatility, Random Forest is a machine learning technique that is frequently used for classification and regression applications. One of Random Forest's primary benefits is its ability to handle high-dimensional data with complex variable interactions. Unlike single decision trees, Random Forest builds multiple decision trees and aggregates their predictions using ensemble averaging, which lowers the risk of overfitting and improves generalization performance. Furthermore, Random Forest is appropriate for datasets with incomplete or noisy data since it naturally manages missing values and retains robustness against outliers [25]. Moreover, the algorithm has the capacity to automatically select the most useful features for predictive modeling, which can improve the efficiency and interpretability of the model .

2) Deep Learning models

IV-C2.1 BERT

Bidirectional Encoder Representations from Transformers, or BERT, is a natural language processing (NLP) paradigm that Google has created. Released in 2018, BERT used the strength of bidirectional context in pre-training to transform the area of natural language processing. BERT can understand the context of a word depending on its left and right surroundings in a phrase, unlike earlier models that processed text sequentially in one way. With the help of this bidirectional method, BERT is better able to comprehend the subtleties and complexities of language, which makes a big difference in a variety of NLP tasks, including question answering, named entity identification, text categorization, and more.

BERT's superior capacity to capture sentence context over earlier NLP models is one of its main advantages. Because of its contextual awareness, BERT is able to construct text representations that are more accurate and perform better on a variety of natural language processing tasks. Moreover, task-specific data may be used to refine BERT's pre-trained representations, which greatly expands its applicability across many domains and applications. Additionally, BERT's open-source design has helped it become widely used and advanced NLP research and development [26].

IV-C2.2 RoBERTa

One kind of machine learning model intended for natural language processing (NLP) is called RoBERTa. It is an enhanced variant of BERT, or Bidirectional Encoder Representations from Transformers, a programme well-known for its capacity to comprehend and produce natural language. "Robustly optimised BERT approach" is what RoBERTa stands for. It was created by Facebook AI and expands on the BERT model

by optimising the training procedure and implementing further changes to boost efficiency.

By optimising the pre-training phase, enhancing comprehension of linguistic subtleties, and surpassing BERT on several benchmarks, RoBERTa improves NLP tasks. It is more accurate at tasks like question answering, text categorization, and language production. Additionally, training on a bigger dataset improves the resilience of RoBERTa [27].

V. Results and Discussion

A. Results

In our study, all models performed exceptionally well, achieving over (90%) accuracy on both validation and test datasets. The decision tree model, while the lowest performer, still achieved a solid (93–94%) accuracy. The logistic regression model was the best, with an impressive (99%) accuracy. Other models like SVM and random forest were also very close, with accuracies between (97-99%) on the test dataset. Even though SVMs are not always the best for large datasets, they worked well in our case because we split the training data into four sets.

For natural language processing, RoBERTa slightly outperformed BERT, with about a (1%) higher accuracy on both validation and test datasets. Additionally, the ensemble voting method achieved (98%) accuracy on both datasets, indicating strong generalisation and avoiding overfitting, even before training on the validation dataset.

The table below shows the results of the RoBERTa pre-trained model with various ML models:

Model	Validation Accuracy	Test Accuracy
Logistic Regression	0.9902	0.9920
DT	0.94545	0.9465
SVM	0.9847	0.9879
RF	0.9783	0.9811

The table below shows the results of the BERT pre-trained model with various ML models:

Model	Validation Accuracy	Test Accuracy
Logistic Regression	0.9863	0.9884
DT	0.9367	0.9369
SVM	0.9854	0.9877
RF	0.9692	0.9744

To enhance generalization and prevent overfitting, we added a voting ensemble step for eight model predictions, achieving approximately 99% accuracy on both validation, and test datasets, as shown below:

Dataset	Accuracy
Validation	0.9833
Test	0.9866

All resources and work are available on GitHub here

```
print("\nEnsemble Voting on Validation dataset:\n")
ensemble_val_predictions = voting(val_results, val_df)

print("\nEnsemble Voting on Test dataset:\n")
ensemble_test_predictions = voting(results, test_df)
```

```
Ensemble Voting on Validation dataset:

Roberta Logistic Regression Accuracy: 0.9902431564459397
Roberta Decision Tree Accuracy: 0.9454962532497324
Roberta SVM Accuracy: 0.9847071417647958
Roberta Random Forest Accuracy: 0.9783758984554213
Bert Logistic Regression Accuracy: 0.9863893561706683
Bert Decision Tree Accuracy: 0.936779324055666
Bert SVM Accuracy: 0.985471784676556
Bert Random Forest Accuracy: 0.9692307692307692
Ensemble Accuracy: 0.9833307845236274

Ensemble Voting on Test dataset:

Roberta Logistic Regression Accuracy: 0.9920173721556154
Roberta Decision Tree Accuracy: 0.9465378027893321
Roberta SVM Accuracy: 0.9879190114998777
Roberta Random Forest Accuracy: 0.9811291901149988
Bert Logistic Regression Accuracy: 0.9884389527770981
Bert Decision Tree Accuracy: 0.9369035967702471
Bert SVM Accuracy: 0.9877660875948128
Bert Random Forest Accuracy: 0.9744922926351848
Ensemble Accuracy: 0.98660386591632
```

Fig. 5. Accuracy of SQL Injection Detection and voting results on validation and test dataset.

B. Discussion

In this study, we used BERT and RoBERTa pre-trained models and found that RoBERTa performed a bit better than BERT. However, we could try other deep learning models to see if they give us even better accuracy. We also thought about trying other machine learning models or sticking with the best ones we found, like logistic regression and random forest.

Our approach relies mostly on SQL queries and labels; however, additional features could improve our ability to identify SQL injection vulnerabilities in Internet of Things devices. Other kinds of IoT attacks could be identified by our approach.

We got our best results, nearly (99%) accuracy, with a medium-sized dataset. Testing our models on larger datasets might give us more insights and possibly better performance. This shows there is still room to improve, and further research could lead to even better solutions for IoT security.

VI. Conclusion

In our study in SQL injection detection , we achieved (99%) accuracy on the test dataset by using a voting ensemble method, which helped us avoid overfitting. However, the size of the dataset is still an important factor. We got great results with a medium-sized dataset, but future work should test our models on larger datasets to see if we can improve even more.

Using a mix of deep learning and machine learning models was very helpful. This hybrid approach combined the strengths of both types of models, leading to strong and accurate detection of vulnerabilities. This means that using hybrid models in future research could provide even better solutions for IoT security.

REFERENCES

[1] V. Sachidananda, S. Bhairav, and Y. Elovici, "Over: Overhauling vulnerability detection for iot through an adaptable and automated static analysis framework," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, 2020, pp. 729–738.

[2] K. M. Sadique, R. Rahmani, and P. Johannesson, "Towards security on internet of things: Applications and challenges in technology," *Procedia Computer Science*, vol. 141, pp. 199–206, 2018, the 9th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2018) / The 8th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2018) / Affiliated Workshops. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918318180

[3] H. Sun, Y. Du, and Q. Li, "Deep learning-based detection technology for sql injection research and implementation," *Applied Sciences*, vol. 13, no. 16, p. 9466, 2023.

[4] L. Song and M. García-Valls, "Improving security of web servers in critical iot systems through self-monitoring of vulnerabilities," *Sensors*, vol. 22, no. 13, p. 5004, 2022.

[5] C. P. Kaliappan, K. Palaniappan, D. Ananthavadivel, and U. Subramanian, "Advancing iot security: a comprehensive ai-based trust framework for intrusion detection," *Peer-to-Peer Networking and Applications*, pp. 1–21, 2024.

[6] R. C. Goswami, H. Joshi, and S. Gautam, "Artificial intelligence-based internet of things security," in *Machine Learning and Metaheuristics: Methods and Analysis*. Springer, 2023, pp. 199–214.

[7] M. Malik, M. Dutta *et al.*, "Contiki-based mitigation of udp flooding attacks in the internet of things," in *2017 International Conference on Computing, Communication and Automation (ICCCA)*. IEEE, 2017, pp. 1296–1300.

[8] M. Dutta, J. Granjal *et al.*, "Towards a secure internet of things: A comprehensive study of second line defense mechanisms," *IEEE Access*, vol. 8, pp. 127 272–127 312, 2020.

[9] A. M. Zarca, J. B. Bernabe, R. Trapero, D. Rivera, J. Villalobos, A. Skarmeta, S. Bianchi, A. Zafeiropoulos, and P. Gouvas, "Security management architecture for nfv/sdn-aware iot systems," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8005–8020, 2019.

[10] S. O. Uwagbole, W. J. Buchanan, and L. Fan, "Applied machine learning predictive analytics to sql injection attack detection and prevention," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2017, pp. 1087–1090.

[11] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain, "Machine learning-based network vulnerability analysis of industrial internet of things," *IEEE internet of things journal*, vol. 6, no. 4, pp. 6822–6834, 2019.

[12] A. Samy, H. Yu, and H. Zhang, "Fog-based attack detection framework for internet of things using deep learning," *Ieee Access*, vol. 8, pp. 74 571–74 585, 2020.

[13] A. Makiou, Y. Begriche, and A. Serhrouchni, "Hybrid approach to detect sqli attacks and evasion techniques," pp. 452–456, 2014.

[14] A. Ghafarian, "A hybrid method for detection and prevention of sql injection attacks," pp. 833–838, 2017.

[15] S. Bharati and P. Podder, "Machine and deep learning for iot security and privacy: applications, challenges, and future directions," pp. 1–41, 2022.

[16] S. S. H. Shah, "Sql injection dataset," https://www.kaggle.com/datasets/syedsaqlainhussain/sql-injection-dataset.

[17] M. S. Abu Syeed Sajid Ahmed, "Sql injection dataset," https://www.kaggle.com/datasets/sajid576/sql-injection-dataset.

[18] W. Yu, "Biggest sql injection dataset," https://www.kaggle.com/datasets/gambleryu/biggest-sql-injection-dataset.

[19] M. Ghaemi, "Sql injection dataset," https://www.kaggle.com/datasets/mahdighaemi/sql-injection.

[20] K. Salah, "Sql injection dataset," https://www.kaggle.com/datasets/kholoodsalah/sql-injection-dataset.

[21] Alextrinity, "Sql injection extend dataset," https://www.kaggle.com/datasets/alextrinity/sqlinjectionextend.

[22] P. Peduzzi, J. Concato, E. Kemper, T. R. Holford, and A. R. Feinstein, "A simulation study of the number of events per variable in logistic regression analysis," *Journal of Clinical Epidemiology*, vol. 49, no. 12, pp. 1373–1379, 1996.

[23] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*. John Wiley & Sons, 2013, vol. 398.

[24] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*. CRC press, 1984.

[25] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts, "Understanding variable importances in forests of randomized trees," in *Advances in neural information processing systems*, 2014, pp. 431–439.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Bidirectional encoder representations from transformers," *arXiv preprint arXiv:1810.04805*, 2018.

[27] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert approach," *arXiv preprint arXiv:1907.11692*, 2019.