CROSS-SITE SCRIPTING (XSS) ATTACK DETECTION

By Group-8

A VENKATA SATYA -CB.EN.U4AIE22005
K N LAKSHMI -CB.EN.U4AIE22023
K HEMAVARDHAN REDDY -CB.EN.U4AIE22026
K V VAMSHIDHAR REDDY -CB.EN.U4AIE22028

What is XSS Attack?

- A cross-site scripting (XSS) attack injects malicious code into vulnerable web applications. XSS does not target the application directly. Instead, XSS targets the users of a web application.
- A successful XSS attack can cause reputational damages and loss of customer trust, depending on the scope of the attack.

Real-Life Examples of Cross-Site Scripting Attacks

1. British Airways XSS Attack (2018)

Exploitation of XSS Vulnerability

- Magecart hackers targeted a JavaScript library (Feedify) on the British Airways website
- Attackers injected malicious scripts via an XSS vulnerability to redirect customer data

Impact of the Breach

- Skimmed credit card details from 380,000 booking transactions
- Used a fake server with an SSL certificate, mimicking a secure domain, before detection

2. Fortnite XSS Vulnerability (2019)

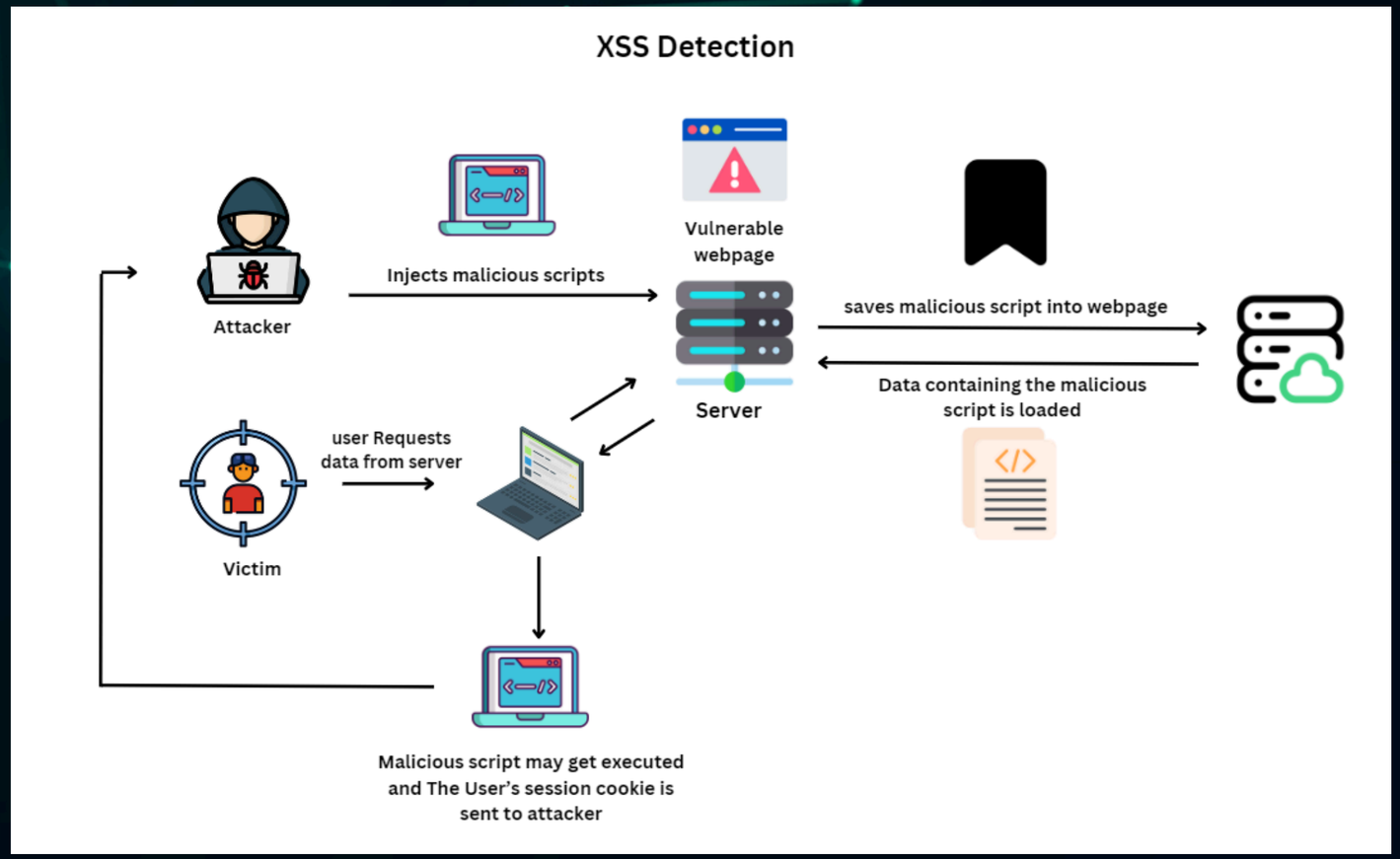
Exploitation of XSS Vulnerability

- Unsecured, retired page with XSS vulnerability affected over 200 million Fortnite users
- Allowed unauthorized access to user data due to oversight by developers

Potential Impact of the Breach

- Could redirect users to a fake login page via XSS and insecure SSO vulnerability
- Enabled theft of virtual currency and recording of player conversations for future attacks

How Does a Cross-Site Scripting Attack Work?



Problem Statement

1) The Threat of XSS Attacks

- Cross-Site Scripting (XSS) is a critical web security vulnerability that allows attackers to inject malicious scripts into trusted websites. These scripts result in data theft, session hijacking, phishing, and unauthorized manipulation of website content.

2) Limitations of Traditional Detection Methods

- Conventional rule-based approaches, such as Web Application Firewalls (WAFs), often struggle to detect sophisticated or obfuscated XSS payloads due to their static nature. Meanwhile, autonomous machine learning models may fail to generalize across diverse XSS patterns, especially when trained on imbalanced or limited datasets, leading to missed detections and false negatives.

Objective

1) Develop a Hybrid Detection System

- Combine a BiLSTM-based deep learning model with regex-based rule matching
- Leverage both pattern recognition and sequence modeling for robust XSS detection

2) To Achieve High Accuracy and Explainability

- Target accuracy exceeding 97%
- Provide confidence scores and reasoning for predictions

3) Ensure Deployable Solution

- Integrate with FastAPI backend and HTML frontend
- Enable real-time detection in web applications

4) Validate on Real-World Data

- Test on a dataset of 27,388 samples (12,263 safe, 14,725 XSS)
- Demonstrate practical applicability and reliability

DATA SET USED

Cross site scripting XSS dataset for Deep learning

Source : Kaggle

Link:

Size and Composition:

- Total Samples: 27,388
- Safe: 12,263 (Class 0)
- XSS Attacks: 14,725 (Class 1, ~53.8%)

Structure:

- Columns: Sentence (HTML/JavaScript snippets), Label (0 = safe, 1 = XSS)
- Filtered to focus on text and label data

Understanding XSS Patterns

Common Patterns:

- <script>alert('xss')</script>: Direct script injection
- <img src=x onerror=alert('xss')>: Event handler exploitation
- javascript:alert(1): Malicious URL scheme

Obfuscated Variants:

- <script>alert('xss')</script>: HTML entity encoding
- \x3Cscript\x3Ealert('xss')\x3C/script\x3E: Hexadecimal encoding

Characteristics:

- Embedded in HTML, JavaScript, or URLs
- Includes event handlers (e.g., onload, onerror)

METHODOLOGY

1)Model Training:

a)Data Preparation: Load XSS_dataset.csv, lowercase sentences, tokenize (10,000 vocab), pad sequences to 200 tokens.

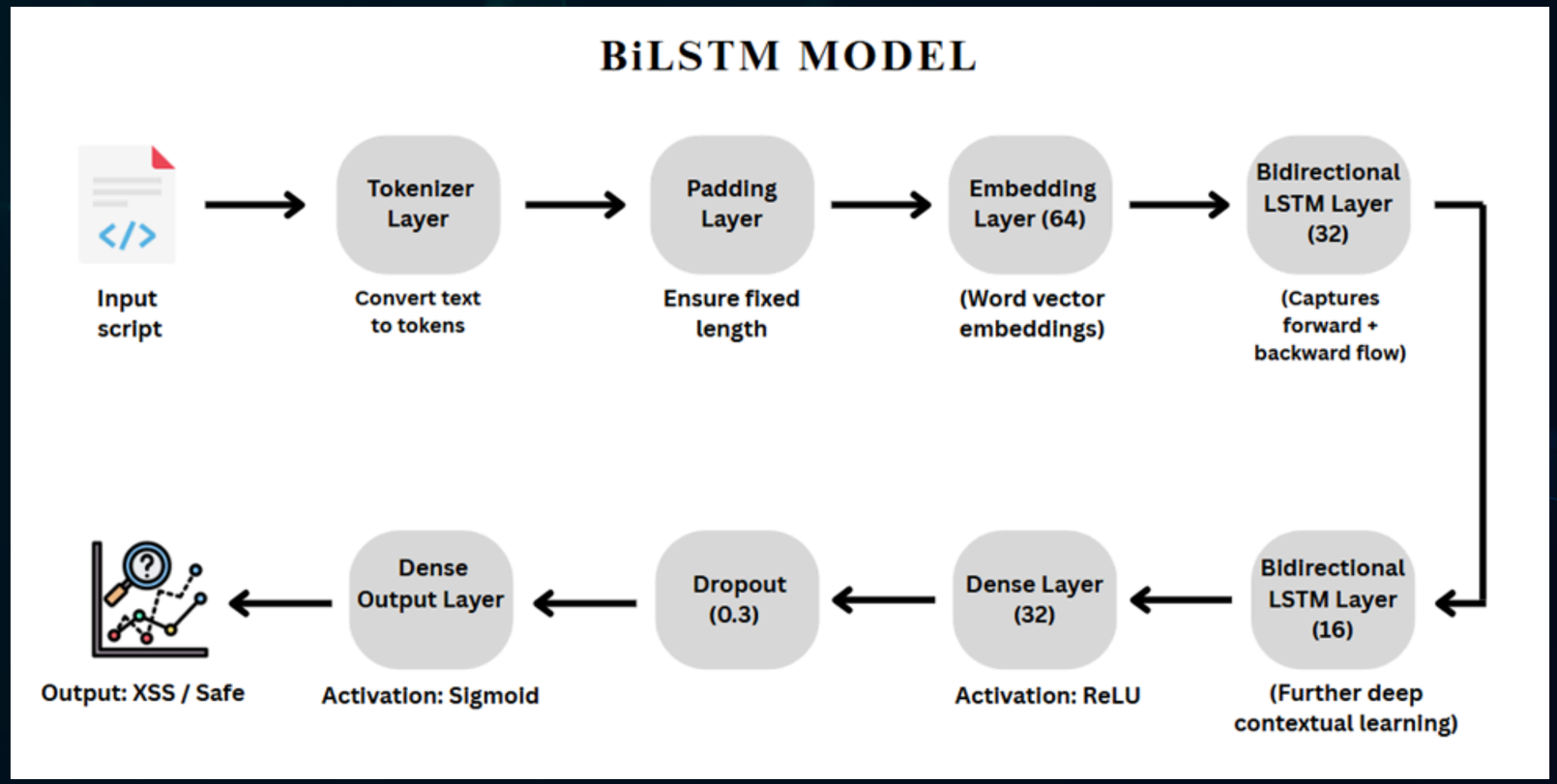
b)Model Architecture: Sequential BiLSTM with:

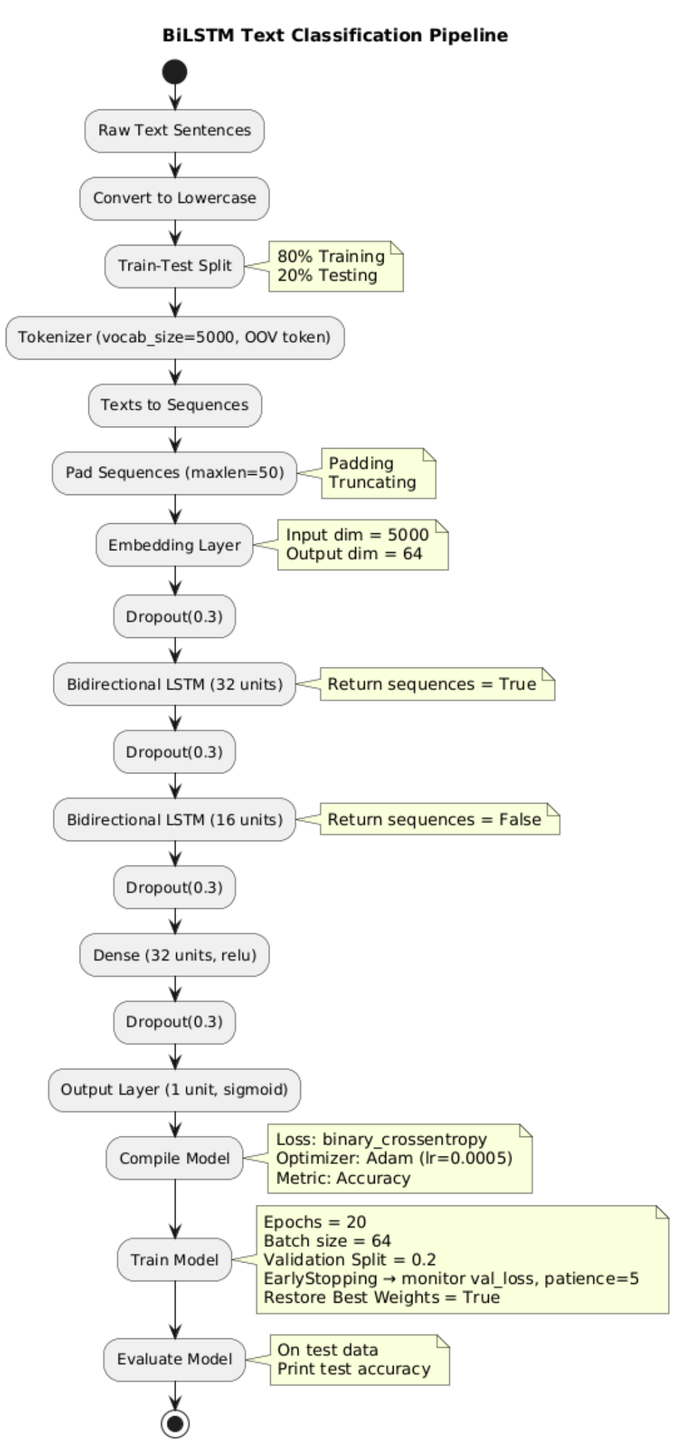
- Embedding (10,000 vocab, 64 dimensions).
- BiLSTM (64 units, return sequences) with dropout (0.3).
- BiLSTM (32 units).
- Dense (32 units, ReLU, L2 regularization).
- Dense (1 unit, sigmoid).

c)Training: Binary cross-entropy loss, Adam optimizer (0.001 learning rate), 10 epochs, early stopping (patience=3), batch size=32, 20% validation split.

d)Evaluation: Test accuracy , classification report, confusion matrix, and loss/accuracy plots.

BLOCK DIAGRAM





METHODOLOGY

2)Backend (FastAPI):

a)XSSDetector Class: Loads the BiLSTM model and tokenizer, defines safe tags/attributes and XSS patterns (e.g., <script>, javascript:).

b)Prediction: Rule-based check first, followed by BiLSTM prediction if inconclusive, returning prediction (0/1), confidence, and reason.

c)API: /predict endpoint handles POST requests with JSON input.

3)Frontend (HTML):

a)UI: Responsive design with textarea, analyze button, and result area (confidence bar).

b)Logic: JavaScript fetches predictions, displays results with color-coded feedback.

How XSS Patterns Are Detected

1. Rule-Based Detection:
   - Uses regex (e.g., <script>, javascript:)
   - Checks safe vs. malicious tags/attributes
2. Deep Learning Detection:
   - BiLSTM analyzes context over 200 tokens
   - Trained on 27,388 samples, 0.9971 accuracy
3. Hybrid Approach:
   - Combines rules and ML for confidence scores

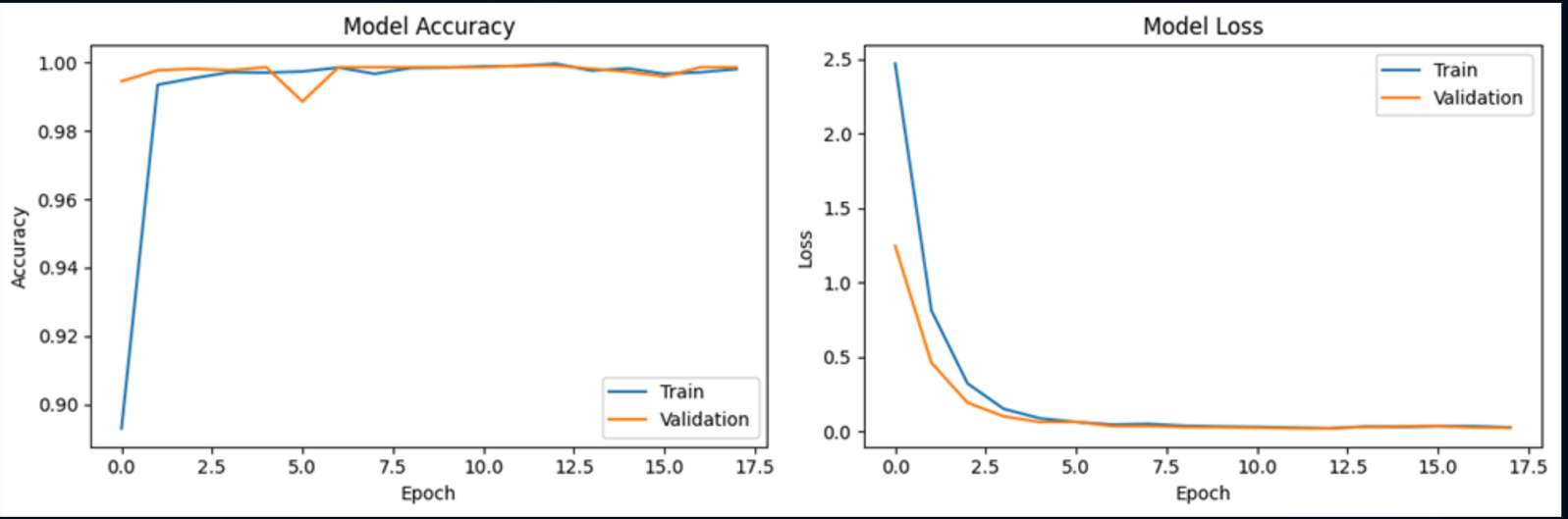
Results

1)Model achieved high accuracy on unseen data (Test Accuracy: 0.9971).

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1263
           1       1.00      1.00      1.00      1475

    accuracy                           1.00      2738
   macro avg       1.00      1.00      1.00      2738
weighted avg       1.00      1.00      1.00      2738


Confusion Matrix:
[[1258    5]
 [   3 1472]]
```

1) Class 0 (safe): Precision 1.00, Recall 1.00, F1-score 1.00, Support 1,258.
2) Class 1 (XSS): Precision 1.00, Recall 1.00, F1-score 1.00, Support 1,472.
3) Accuracy: 1.00, Macro Avg: 1.00, Weighted Avg: 1.00 (over 2,738 samples).
4) Confusion Matrix: 5 false negatives, 3 false positives, showing minimal errors.

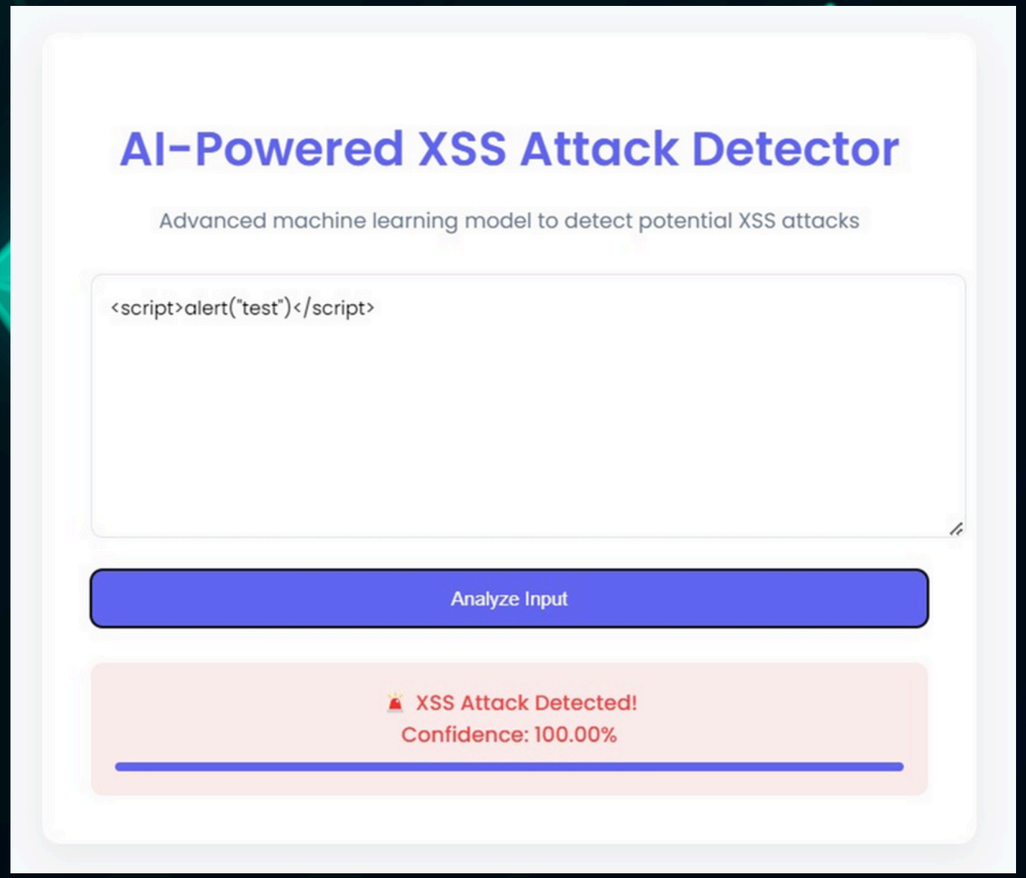
Results



a) Accuracy: Train and validation accuracy stabilize at ~1.00 after 5 epochs, with minor fluctuations.

b) Loss: Train and validation loss drop sharply to ~0.0 by 5 epochs, remaining stable, confirming convergence.

Web Application Integration



1. Backend: "FastAPI on 127.0.0.1:8000, /predict endpoint."
2. Frontend: "HTML interface with confidence bar."
3. Benefit: "Real-time, user-friendly detection."

How to Be Safe from XSS

Filter Input on Arrival

- Check and remove suspicious data at entry
- Example: <script>alert('XSS')</script> → alerthack
- Prevents bad data from processing

Encode Data on Output

- Transform data to text (HTML, JS, CSS, URLs)
- Example: <script> → <script>
- Stops execution even if data is malicious

Use Appropriate Response Headers

- Set Content-Type and X-Content-Type-Options
- Ensures browsers handle responses safely
- Works for simple, non-HTML responses

Content Security Policy (CSP)

- Restricts trusted script sources
- Example: Blocks evil.com scripts
- Last defense against breaches

Conclusion

- This project provides a very efficient XSS detection system that uses a BiLSTM model and rule-based improvements to achieve a test accuracy of 0.9971.
- Real-time, easily available predictions are provided via the HTML frontend and FastAPI backend, which are verified by consistent loss/accuracy curves and flawless classification metrics.
- By addressing issues like latency and overfitting, the system performs better than conventional techniques and provides a deployable web security solution.
- Small problems (such scalability and dataset origin) point to areas that need work.

Future Work

Ideas: "Dataset expansion, transformer models, real-time integration."

Goals: "Improve scalability and handle obfuscation."

References

1) Detection of cross-site scripting (XSS) attacks using machine learning techniques
https://link.springer.com/article/10.1007/s10462-023-10433-3

2) Machine and Deep Learning-based XSS Detection Approaches
https://www.sciencedirect.com/science/article/pii/S1319157823001829

3) An Efficient XSS Attack Detection Approach Based on Machine Learning Classifier on SNSs
https://koreascience.kr/article/JAKO201726163355323.page

THANK YOU