Vol.2021-CSEC-95 No.11
Vol.2021-SPT-45 No.11
Vol.2021-EIP-94 No.11
2021/11/8

ONLSTMとAttention機構によるXSS攻撃の検知に関する一考察

宮崎　裕一郎[†1,a)]　三村　守[†1,b)]

概要：ウェブアプリケーションに対するXSS攻撃は，2020年度第3四半期にIPAに提出された届出の約58 %を占めている．これに対し，スキーム，ホスト名，ドメイン名などから，機械学習モデルを用いてXSS攻撃を検知する手法が提案されている．しかしながら，これらの研究の評価に用いたデータセットには，実際のXSS攻撃の特徴が含まれておらず実用的とは言い難い．そこで本研究では，攻撃の特徴が表れるディレクトリやパラメータに着目し，実際のXSS攻撃の特徴を検知することを試みた．さらに，Attention機構を用いて注目した特徴を視覚化することで，モデルの検知原理の解明を試みた．本研究では，Attention機構とONLSTM (Ordered-Neurons Long Short-Term Memory) を組み合わせた検知モデルを提案する．検知精度の評価は，XSSed.comおよびCIC-IDS2017データセットから45,656体のサンプルを抽出し，ディレクトリやパラメータを含む不均衡なデータセットを作成して実施した．その結果，提案手法ではF値0.98の高精度でXSS攻撃を検知することを確認した．さらに，Attention機構を用いることで，提案手法がXSS攻撃の特徴に基づいて検知していることを確認した．

キーワード：XSS，Attention機構，ONLSTM

1. はじめに

IPAによる報告書「ソフトウェアなどの脆弱性関連情報に関する届出報告」によると，2020年第3四半期に報告された脆弱性のうち約58 %をクロスサイトスクリプティング（XSS）が占めていた．これらの攻撃による被害には，フィッシング，セッションハイジャック，ウェブサイトの改ざん等がある．その運用面の対策としては，入力値の制限，サニタイジング（スクリプトの無効化）等が挙げられる．研究分野での対策としては，機械学習を用いたXSS攻撃の検知手法が検討されている．しかしながら，従来の研究では実用性が十分に考慮されているとは言い難い．たとえば，一部の研究ではXSSの対象がJavascriptに限定されている [1], [2]．また，データセットの構成が実用的ではない [1], [2], [3], [4]，サンプル数が十分ではない [1], [5], [6] 等の制約もある．

機械学習モデルの精度を評価するにあたっては，データセットの構成に注意する必要がある．たとえば，文献 [1], [2] では，正常入力のデータセットとしてDMOZ data[*1]およびThe clueweb09 dataset [*2]を使用しているが，これらのデータセットにはURLの末尾にパラメータが含まれているものが存在しない．そのため，パラメータの有無だけで容易に正常入力と判定することが可能であると考えられる．文献 [3] では，正常入力のデータセットとして [7] から収集したデータを使用している．このデータには，パラメータおよびディレクトリが含まれているものが存在しない．そのため，入力の長さのみで容易に正常入力と判定することが可能であると考えられる．このように，従来の研究ではモデルが実際にXSS攻撃の特徴をとらえているかどうかは十分に検証されていない．このようなモデルの評価に用いられるデータセットが不適切であるという課題は，文献 [8] でも報告されている．本研究では，フォームに攻撃入力を行った際のパラメータと正常入力の相違に注目する．

本研究ではXSS攻撃を検知するために，BiLSTM (Bidirectional Long Short-Term Memory) とAttention機構，およびONLSTM (Ordered-Neurons Long Short-Term Memory) とAttention機構を組み合わせたモデルを提案する．これらのモデルは，自然言語処理における文章分類におい

[†1] 防衛大学校
National Defense Academy
[a)] s65693@nda.ac.jp
[b)] mim@nda.ac.jp

[*1] https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/OMV93V
[*2] https://lemurproject.org/clueweb09/

て高い精度を示しており [9], [10]，正常入力と XSS 攻撃の分類にも応用できると考えられる．これまでの研究でも，XSS 攻撃を検知するために様々なニューラルネットワークを用いたモデルが提案されている [3], [4], [11]．しかしながら著者らが知り得る限り，これらのモデルを適用した研究はこれまでに存在しない．

さらに本研究では，モデルが実際に XSS 攻撃の特徴をとらえているかどうかを確認するために，Attention 機構を用いて注目した特徴を視覚化して分析する．これまでにも Attention 機構を使用した手法は提案されている [11]．しかしながら，モデルの検知原理は十分に検証されているとは言い難い．

本研究における貢献は以下のとおりである．

(1) 正常入力として，ディレクトリやパラメータが含まれる実用的なデータセットを使用した．

(2) BiLSTM と Attention 機構および ONLSTM と Attention 機構を組み合わせた XSS 攻撃の検知手法を提案した．

(3) Attention 機構の注目箇所を分析し，モデルが XSS 攻撃を検知する原理を検証した．

2. XSS の分類と対策

2.1 XSS の分類

XSS 攻撃とは，XSS の脆弱性を悪用した攻撃である．CWE-79 [12] によると，XSS の脆弱性は反射型 XSS，格納型 XSS，DOM-BasedXSS の 3 つに分類される．

反射型 XSS は，ユーザからのリクエストに含まれる悪性スクリプトを，ウェブアプリケーションがそのリクエストへのレスポンスのウェブページに，実行可能なスクリプトとして出力する XSS の脆弱性である．そのスクリプトがリクエストである送信者に戻るため，反射型 XSS と呼ばれている．

格納型 XSS は，ユーザからのリクエストに含まれる悪性スクリプトを，ウェブアプリケーションの内部に恒久的に保存することで，当該ウェブページに実行可能なスクリプトとして出力する XSS の脆弱性である．この脆弱性があるページをユーザが閲覧する度にスクリプトが実行されるため，格納型 XSS と呼ばれている．

DOM-BasedXSS は，ウェブページに含まれる正規のスクリプトを利用して動的にウェブページ使用した結果，本来想定されていないスクリプトをウェブページに出力してしまう XSS の脆弱性である．ウェブページを操作する機能である DOM (Document Object Model) を悪用したものであることから，DOM-Based XSS と呼ばれている．

2.2 XSS 攻撃への対策

XSS 攻撃への運用面の対策としては，ウェブアプリケーションにおける入力値の制限，サニタイジング等が挙げられる．また，入力された HTML テキストから構文解析木を作成し，必要な要素のみを抽出するという手法もある．他には，HTTP レスポンスヘッダに存在する Content-Type フィールドに，文字コードを指定するという対策も挙げられる．研究分野での対策としては，ニューラルネットワークや機械学習モデルを利用して XSS 攻撃を検知する手法が検討されている．

3. 関連研究

XSS 攻撃を検知するために，機械学習やニューラルネットワークを用いた手法が提案されている．

文献 [3] では，多層パーセプトロンを用いた手法が提案されている．この研究では，正常入力のデータセットとして Alexa-static [7] を使用している．このデータセットには，パラメータやディレクトリが含まれているものが存在せず，その特徴量として URL の長さを使用している．しかしながら，パラメータやディレクトリを含まない正常入力は，攻撃入力より長さが短くなるのは自明である．そのため，モデルの動作原理については検証の余地がある．

文献 [4] では，LSTM (Long Short-Term Memory) を用いたモデルを提案している．文献 [11] では，LSTM と Attention 機構を組み合わせたモデルを提案している．これらの研究では，正常入力のデータセットとして DMOZ data を使用している．このデータセットには，URL にパラメータが含まれていない．そのため，正常入力の特徴を十分に代表しているとは言い難い．URL にパラメータを含む正常なウェブサイトにおいては，誤検知が発生してしまう可能性がある．

文献 [1] では，SVM，k 近傍法，および Random Forest を使用して，XSS 攻撃に使用される JavaScript のコードを検知する手法が提案されている．文献 [13] では，CNN (Convolutional Neural Network) と LSTM を組み合わせたモデルを提案している．これらの研究では，JavaScript を用いた XSS 攻撃に対象が限定されている．本研究では，JavaScript を用いない XSS 攻撃も対象としている．

文献 [5] では, SVM (Support Vector Machine), Random Forest，および SCW (Soft Confidence-Weighted) を使用して XSS 攻撃を検知する手法を提案している．攻撃入力 915 体および正常入力 1290 体から，それぞれ 500 体を抽出してデータセットとして使用している．文献 [6] では，SVM および n-gram を用いたモデルを提案している．データセットの規模は，訓練データ，テストデータを合わせて 400 体である．これらの研究では，評価に使用したデータセットの規模が十分とは言い難い．

本研究と関連研究の違いを**表 1** に示す．本研究では，

Vol.2021-CSEC-95 No.11
Vol.2021-SPT-45 No.11
Vol.2021-EIP-94 No.11
2021/11/8

BiLSTM と Attention 機構，および ONLSTM と Attention 機構を組み合わせたモデルを提案する．さらに，ディレクトリやパラメータが含まれる十分な数のデータセットを使用して精度を評価する．既存の研究では，これらのモデルを使用したモデルは提案されていない．また，Attention 機構を使用した手法は提案されているものの，モデルの検知原理は十分に検証されているとは言い難い．

表 1 関連研究との比較

文献	モデル
[5]	SVM，RamdomForest，SCW
[6]	SVM，n-gram
[3]	多層パーセプトロン
[4]	LSTM
[11]	LSTM&Attention 機構
[13]	CNN&LSTM
[1]	k 近傍法，Random Forest
[2]	単純ベイズ分類器，J48 決定木，SVM
本研究	BiLSTM&Attention 機構
	ONLSTM&Attention 機構

4. 本研究で使用するニューラルネットワーク

4.1 LSTM

まず，RNN (Recurrent Neural Network) について説明する．RNN は，隠れ層を利用し，前の出力を次の入力として取り扱うニューラルネットワークの一種である．RNN の短所の 1 つとして，長い時間軸での情報の利用が困難という勾配消失問題が挙げられる．これを，更新ゲート，関連ゲート，忘却ゲート，および出力ゲートを使用して解決を図ったのが LSTM である．LSTM の構成を**図 1** に示す．

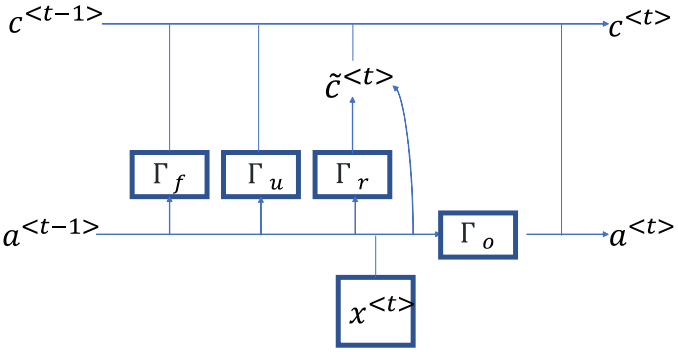

図 1 LSTM のアーキテクチャ

LSTM のゲートは，以下の式で定義される．

$$\Gamma = \sigma\left(Wx^{<t>} + Ua^{<t-1>} + b\right) \tag{1}$$

図中の各セルおよび出力は以下の式で定義される．

$$\tilde{c}^{<t>} = \tanh\left(W_c\left[\Gamma_r \star a^{<t-1>}, x^{<t>}\right] + b_c\right) \tag{2}$$

$$c^{<t>} = \Gamma_u \star \tilde{c}^{<t>} + \Gamma_f \star c^{<t-1>} \tag{3}$$

$$a^{<t>} = \Gamma_o \star c^{<t>} \tag{4}$$

式中の $\star$ はアダマール積を示す．本研究では，LSTM の応用である BiLSTM を使用する．BiLSTM では，前からの入力を処理する Forward LSTM に加え，後ろからの入力も処理する Backward LSTM という機構を備えている．この特徴により，自然言語処理や音声認識において高い精度が報告されている [14].

4.2 ONLSTM

本研究ではさらに，LSTM の応用である ONLSTM [15] も使用する．ONLSTM では，文章の構成を単純な系列ではなく，階層構造で分析する特徴がある．ONLSTM において，忘却すべき領域は以下の式で定義される．

$$\tilde{f}_t = \mathrm{cumax}\left(W_{\tilde{f}}x_t + U_{\tilde{f}}h_{t-1} + b_{\tilde{f}}\right) \tag{5}$$

入力すべき領域は，以下の式で定義される．

$$\tilde{i}_t = 1 - \mathrm{cumax}\left(W_{\tilde{i}}x_t + U_i h_{t-1} + b_{\tilde{i}}\right) \tag{6}$$

(6) の $\tilde{i}_t$ は ONLSTM 中の入力すべき領域である．

$$\omega_t = \tilde{f}_t \circ \tilde{i}_t \tag{7}$$

(7) によって (5)，(6) の重複部分を求めている．

$$\hat{f}_t = f_t \circ \omega_t + \left(\tilde{f}_t - \omega_t\right) \tag{8}$$

(8) の $\hat{f}_t$ は ONLSTM 中の忘却ゲートである．それを通常の LSTM の忘却ゲートの f_t や，(7) によって導出された ω_t を使用して導出している．

$$\hat{i}_t = i_t \circ \omega_t + \left(\tilde{i}_t - \omega_t\right) \tag{9}$$

(9) によって ONLSTM の入力ゲートである $\hat{i}_t$ を，$\tilde{i}_t$ と (11) によって導出された ω_t とノーマルな LSTM の入力ゲートである i_t を使って導出している．

$$\hat{c}_t = \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right) \tag{10}$$

$$c_t = \hat{f}_t \circ c_{t-1} + \hat{i}_t \circ \hat{c}_t \tag{11}$$

新しい情報である $\hat{c}_t$ を (10) によって導出し，(11) によって，ONLSTM の忘却ゲートである $\hat{f}_t$ と ONLSTM の入力ゲートである $\hat{i}_t$ を使用してメモリを更新している．

4.3 Attention 機構

Attention 機構は，Encoder-Decoder モデルに導入される要素ごとの関係性および注意箇所を学習する機構である．一般的な Attention 機構における Encoder-Decoder は (12)

によって表される．この時，Decoder の隠れ層は Target，Encoder の隠れ層は Source と表される．

$$\begin{aligned}&\text{Attention}\left(\text{Target},\ \text{Source}\right)\\&=\text{Softmax}\left(\text{Target}\cdot\text{Source}^{T}\right)\cdot\text{Source}\end{aligned}\tag{12}$$

そして Source を Key と Value に分割し，Decoder の隠れ層を検索クエリ（query）と表現すると，(13) のように，表される．また，この時 Key と Value は辞書オブジェクトとなっている．

$$\begin{aligned}&\text{Attention(query, Key, Value)}\\&=\text{Softmax}\left(\text{query}\cdot\text{Key}^{T}\right)\cdot\text{Value}\end{aligned}\tag{13}$$

Attention 機構は，主に入力されたデータのどこに注意するかによって，自己注意とソースターゲット注意に区分される．つまり，これらの注意の違いは，どこから Query，Key，Value が導出されるかということであり，本研究において使用した自己注意はすべて同じ場所，Encoder において使用される Query，Key，Value ならば，すべて下の隠れ層から導出されるという特徴を持つ．

5. 提案手法

5.1 概要

本研究では，BiLSTM と Attention 機構，および ONLSTM と Attention 機構を組み合わせたモデルにより，XSS 攻撃を検知する手法を提案する．**図 2** に本研究の概要を示す．

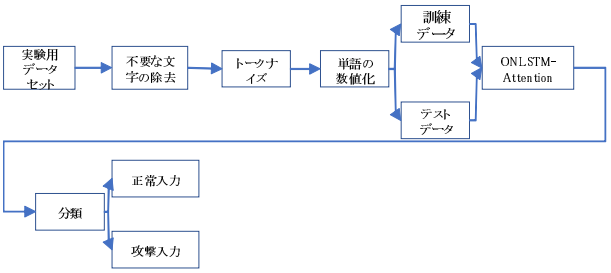

図 2 本研究の概要

本研究では，データセットの前処理を実施した後，訓練データとテストデータに分割する．訓練データを用いて提案モデルを訓練した後，未知のテストデータを入力して攻撃入力か正常入力かを分類する．提案モデルの構成を**図 3** に示す．

提案モデルでは，入力値を分割した単語を入力としている．数値化された単語は，LSTM Layer および Attention Layer の順に処理される．提案モデルの出力は，攻撃入力 (XSS) か正常入力 (Non-XSS) のどちらかとなる．

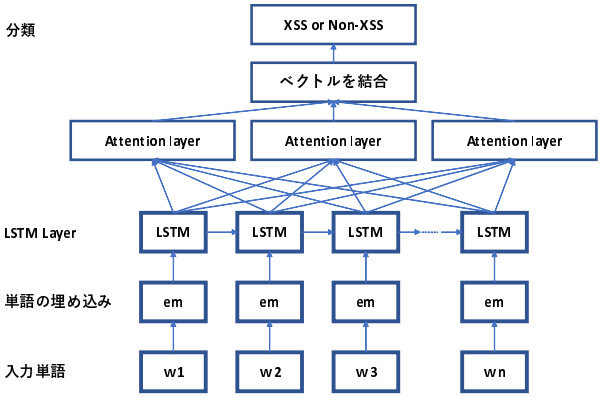

図 3 提案モデルの構成

5.2 前処理

前処理では，入力値から不要な文字を除去して単語に分割（トークナイズ）し，各々を数値に変換する．まず，入力値の「.」および「/」を空白文字に置換し，スキームとホスト名を削除する．次に，空白文字で分割して分かち書きを実施し，それらを事前に 60 億の単語を使用して作成済みの，glove.6B.200d.txt[*3]を使用し，200 次元の単語埋め込みベクトルに変換する．前処理は NLTK (Natural Language Toolkit)-3.2.5 [*4]を用いて実装した．NLTK は，形態素解析のためのプラットフォームであり，構文解析，分かち書き，品詞の取得等の機能が実装されている．

5.3 実装およびパラメータの設定

提案モデルは, PyTorch1-1.9.0 [*5]を使用し, Python-3.7.12 [*6]を用いて実装した．バッチサイズに関しては，epoch 毎の損失関数の振れ幅が一定の範囲に収まるように，40 に設定した．単語の埋め込みのための次元数は，データセットの規模および学習時間を考慮し，200 次元とした．ONLSTM の隠れ層の次元数は，学習時間および過学習の可能性を考慮して 128 次元とし，Dropout を 0.2 に設定した．Attention Layer は 3 層構成とし，BiLSTM と Self-Attention 機構を組み合わせたモデルにおいても同様の構成とした．損失関数は，以下の式で定義される PyTorch の NLLLoss を使用した．

$$l_n = -w_{y_n} x_{n,y_n}\tag{14}$$

式中の x は，データ数を表す N と，クラス数を表す C を用いた，$N \times C$ のテンソルである．y は，N 次元のベクトルとして，$0 < y_n < C$ と定義される．w はデフォルト値の 1 とした．活性化関数は，BiLSTM および ONLSTM の双方においてシグモイド関数および Tanh 関数を使用した．

ONLSTM は，文献 [15] と同様の一般的な構成とした．

*3 https://nlp.stanford.edu/projects/glove/
*4 https://www.nltk.org
*5 https://pytorch.org
*6 https://www.python.org

1層の ONLSTM の各隠れ層の出力を，3 層の Attention Layer に入力する構成とした．

提案モデルでは，Attention 機構の一種の自己注意機構を用いた．一般的な Attention 機構との相違点は，Query，Key，Value がすべて同じ隠れ層から導出されるという点である．query に一致する key を探し，それに対応する value を抽出する．提案モデルでは，各 Attention Layer で ONLSTM の隠れ層から受け取ったベクトルを重み付けし，それを結合したものを用いて二値分類を実施する．

6. 性能評価

6.1 評価指標

本研究では，入力された文字列を攻撃入力だと予測することを Positive，正常入力だと予測することを Negative とする．正常入力と攻撃入力の関係は**表 5** に示すとおりである．

表 2 正常入力と攻撃入力の関係

		実際の結果	
		攻撃入力	正常入力
予測結果	攻撃入力	True Positive (TP)	False Positive (FP)
	正常入力	False Negative (FN)	True Negative (TN)

分類問題の評価指標としては，一般的に以下の 4 種が使用される．

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN} \tag{15}$$

$$\text{Precision} = \frac{TP}{TP+FP} \tag{16}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{17}$$

$$\text{F1} = \frac{2\ \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \tag{18}$$

Accuracy（正解率）は，予測結果と正答がどの程度一致しているかを判断する指標である．今回の実験では，攻撃入力と正常入力の割合が不均衡であるため，Accuracy は使用しない．Precision（適合率) は，攻撃入力と予測したサンプルのうち，実際に攻撃入力であったサンプルの割合である．Recall（再現率）とは，実際に攻撃入力であるサンプルのうち，攻撃入力だと予測したサンプルの割合である．F1（F 値）は，適合率と再現率の調和平均により算出される評価尺度である．今回の実験では，攻撃入力を漏れなく検出できるかを検証するため，Recall および F1 に着目する．

6.2 データセット

提案モデルの精度を評価するためのデータセットは，攻撃入力のサンプルと正常入力のサンプルから構成されている．攻撃入力は，XSSed.com からクローラを用いて収集した．XSSed.com [*7]は，XSS に関連する情報を提供する最大級のオンラインアーカイブである．XSSed.com の入力は再現可能であるため，ラベルの信頼性は高いものと考えられる．XSS Archive というページにリクエストを送り，その中の「mirror」リンクの URL を再帰的にたどって収集した．クローラは Beautiful Soup [16] を用いて実装し，サーバに負荷をかけないよう細心の注意を払って実装した．

正常入力は，Intrusion Detection Evaluation Dataset (CIC-IDS2017) [*8]から抽出した．このデータセットは，カナダのサイバーセキュリティ研究所が，IDS および IPS で 5 日間に渡ってキャプチャしたネットワークトラフィックであり，正常および攻撃のラベルが付与されている [17]．このデータセットから，Benign_list_big_final を正常入力として使用した．このデータセットには，ディレクトリやパラメータも含まれているため，DMOZ data のようにパラメータの有無で容易に正常入力と判定することは困難であると考えられる．

データセットは，概ね 7:3 の比率でランダムに訓練データとテストデータに分割した．他の関連研究においても，概ね同様に比率で分割されている．データセットのサンプル数の内訳を**表 3** に示す．

表 3 データセットのサンプル数

	攻撃入力	正常入力
訓練データ	12569	20000
テストデータ	1024	12063
合計	13593	32063

6.3 実験内容

提案モデル（BiLSTM と Attention 機構，および ONLSTM と Attention 機構を組み合わせたモデル）を訓練データを用いて訓練し，テストデータを用いて精度を評価する．比較対象として，SVM および BoW (Bag-of-Words) を用いたモデルも実装した．実行環境は，Google Colaboratory 無料版であり，CPU は Intel Xeon CPU 2.20-2.30GHz，メモリは 12GB，GPU は NVIDIA Tesla K80 16GB である．

6.4 実験結果

訓練データを用いた場合の各 epoch 毎の損失関数の値の変化を**図 4** に示す．

縦軸は損失関数の値，横軸は epoch 数を示している．ONLSTM では，損失関数の値は epoch 数 20 において十

*7 http://xssed.com
*8 https://www.unb.ca/cic/datasets/ids-2017.html

Vol.2021-CSEC-95 No.11
Vol.2021-SPT-45 No.11
Vol.2021-EIP-94 No.11
2021/11/8

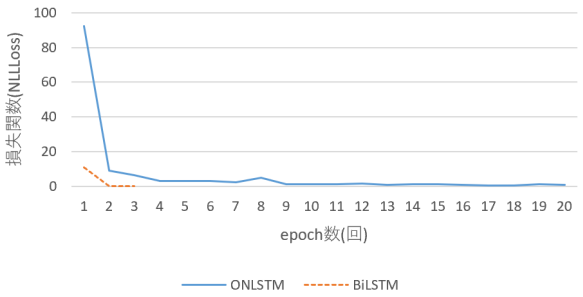

図 4 BiLSTM および ONLSTM の損失関数の値の変化

分に小さい値となった．BiLSTM では，損失関数の値は epoch 数 3 において十分に小さい値となった．モデルの過学習を防止するため，これらの epoch 数において訓練を停止し，テストデータを入力してモデルの精度を評価した．

表 4 に提案モデルおよび SVM と BoW を用いたモデルの精度を示す．提案モデルでは，正常入力 (Non-XSS) に対する Precision，Recall，F1 はすべて高い精度となった．ONLSTM では，攻撃入力 (XSS) に対する Recall と F1 はそれぞれ 0.97 と 0.98 であり，やや見逃しが発生することが確認できた．提案モデルの ONLSTM を BiLSTM に置き換えた場合，Recall および F1 は向上し，それぞれ 0.99 および 1.00 となった．SVM と BoW を用いたモデルでは，正常入力に対する Precision，Recall，F1 は概ね良好であるものの，攻撃入力はほとんど検知できていないことが確認できた．

表 4 提案モデルおよび SVM と BoW を用いたモデルの精度

		Precision	Recall	F1
BiLSTM &Attention	Non-XSS	1.00	1.00	1.00
	XSS	1.00	0.99	1.00
ONLSTM &Attention	Non-XSS	1.00	1.00	1.00
	XSS	1.00	0.97	0.98
BoW&SVM	Non-XSS	0.96	0.98	0.97
	XSS	0.64	0.49	0.55

7. 考察

7.1 分類精度

実験の結果，提案モデルでは正常入力はほぼ完璧に分類できることが確認できた．攻撃入力に対してはやや見逃しが発生するものの，ほぼ正確に分類できることが確認できた．これに対し，SVM と BoW を用いた単純なモデルでは，攻撃入力はほとんど検知できていないことが確認できた．この結果から，データセットの構成は適切であり，単純なモデルでは攻撃入力の検知は困難であることが確認できる．提案モデルでは，同じデータセットを用いたにもかかわらず，ほぼ正確に攻撃入力を分類することが可能であった．したがって，データセットの構成は適切であり，提案モデルは高精度で攻撃入力を分類することが可能であると考えられる．

7.2 分類に貢献した特徴

分類の貢献した特徴を分析するために，Attention 機構で注意した箇所を視覚化して分析した．視覚化した特徴の例を**図 5**，**図 6**，および**図 7** に示す．

図 5 BiLSTM+Attention 機構の注意箇所

図 6 ONLSTM+Attention 機構の注意箇所

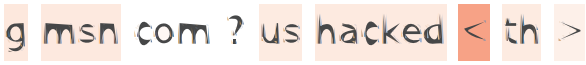

図 7 ONLSTM+Attention 機構の注意箇所 2

これらはいずれも攻撃入力であり，提案モデルによって攻撃入力と分類された文字列である．図中の色が濃いほど，注意の度合いが強いことを示している．図 5 および図 6 では，文字列の後半の単語全般に注意しているのに対し，図 7 では特定の単語や記号に注意していることが確認できる．これらの傾向は，他の入力においても確認されている．このように「¡th¿」などの HTML タグに注意しているのは，現実のウェブサイトにおいて HTML タグの入力を禁止しているフォームが多いことに起因しているものと考えられる．また，他の文字列に関しても同様に，入力フォームで禁止している特徴的な文字である「¡」「¿」や「/」などに注意していることが確認できた．図 5 のように，XSS のテストのために入力する「hacked」という文字列や，「script」，「alert」といった様々な文字列への注意も認められた．これは，XSSed.com から収集した攻撃入力に，不特定多数の人物がテストのために使用する様々な文字列が含まれていたためであると考えられる．しかしながらこの場合にも，実際の XSS のために使用される HTML タグや，前述の記号に注意している場合が多く認められた．このように，提案モデルでは実際に XSS 攻撃の特徴をとらえているものと考えられる．

7.3 他の研究との比較

表 5 に精度およびサンプル数の比較を示す．

本研究では，十分な数のサンプルを用いており，良性サンプルの割合が多い不均衡なデータセットを用いて評価を実施している．さらに，正常入力にはパラメータやディレクトリが含まれているため，単純に URL の長さやパラ

表 5　精度およびサンプル数の比較

文献	Recall	F1	攻撃入力数	正常入力数
[5]	0.988	0.992	500	500
[6]	0.99	未記載	400	未記載
[3]	0.9835	0.9877	38569	100000
[4]	0.939	0.939	33426	31407
[11]	0.982	0.985	32168	78652
[13]	0.991	0.995	40000	74000
[1]	0.9961	未記載	合計で 15000	
[2]	未記載	未記載	44264	19646
本研究	0.97	0.98	13087	32569

メータの有無で分類することはできない．それにもかかわらず，提案モデルでは既存研究と同程度の高精度で攻撃入力を分類することが可能であった．

7.4　研究倫理

本研究では，クローリングを含むすべての実験を Google Colaboratory 上で実施した．したがて, 一般的なコンピュータでもウェブサイトにアクセスできる環境であれば，再現可能である．クローラのソースコードおよびデータセットはレポジトリ [18] に公開し，再現性を確保している．

本研究を実施するに当たり，倫理的側面には細心の注意を払った．本研究において，攻撃入力を収集に用いたクローラは，連続アクセスによるウェブサーバの負荷を減らすため，0.1 秒から 0.5 秒の間隔を空けてリクエストを行うように実装した．

7.5　研究の限界

本研究では，攻撃入力として XSSed.com から収集したデータセットを使用した．しかしながらこれは，実際に発生した XSS 攻撃を入力ではない．攻撃入力のデータセットは実際の XSS 攻撃の入力であることが望ましいが，公開されているデータセットからは見つけることはできなかったと結論付けられている [8]．実際の XSS 攻撃の入力を含むデータセットを入手することができれば，提案モデルの実用性を評価することが可能であると考えられる．

8.　おわりに

本研究では, BiLSTM と Attention 機構および ONLSTM と Attention 機構を組み合わせた XSS 攻撃の検知手法を提案し，正常入力としてディレクトリやパラメータが含まれる実用的なデータセットを使用して評価を実施した．さらに，Attention 機構の注目箇所を分析し，モデルが XSS 攻撃を検知する原理を検証した．本研究ではランダムに訓練データを選択したが，訓練データに偏りがある可能性も考えられる．

偏りの影響を緩和するための反復実験や，交差検証については今後の課題である．より大規模なデータセットや，実際の XSS 攻撃の入力を含むデータセットでの評価も今後の課題である．

参考文献

[1] Fawaz A Mereani and Jacob M Howe. Detecting cross-site scripting attacks using machine learning. In *International Conference on Advanced Machine Learning Technologies and Applications*, pp. 200–210. Springer, 2018.

[2] BA Vishnu and KP Jevitha. Prediction of cross-site scripting attack using machine learning algorithms. In *Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing*, pp. 1–5, 2014.

[3] Fawaz Mahiuob Mohammed Mokbal, Wang Dan, Azhar Imran, Lin Jiuchuan, Faheem Akhtar, and Wang Xiaoxi. Mlpxss: an integrated xss-based attack detection scheme in web applications using multilayer perceptron technique. *IEEE Access*, Vol. 7, pp. 100567–100580, 2019.

[4] Yong Fang, Yang Li, Liang Liu, and Cheng Huang. Deepxss: Cross site scripting detection based on deep learning. In *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence*, pp. 47–51, 2018.

[5] 梅原章宏, 松田健, 園田道夫, 水野信也, 趙晋輝. 機械学習を用いたクロスサイトスクリプティング (xss) 攻撃の検知に関する考察. Technical Report 13, 中央大学大学院理工学研究科情報工学専攻, 静岡理工科大学総合情報学部コンピュータシステム学科, 中央大学大学院理工学研究科情報工学専攻, 静岡理工科大学総合情報学部コンピュータシステム学科, 中央大学理工学部情報工学科, nov 2015.

[6] Gulit Habibi and Nico Surantha. Xss attack detection with machine learning and n-gram methods. In *2020 International Conference on Information Management and Technology (ICIMTech)*, pp. 516–520. IEEE, 2020.

[7] Alexa. Alexa-static. `http://s3.amazonaws.com/alexa-static/top-1m.csv.zip`. (Accessed on 10/10/2021).

[8] 飯野和真, 宇田隆哉. 不適切なデータセットや処理方法を用いた機械学習による xss 攻撃検出研究の解説と精度の比較. Technical Report 20, 東京工科大学, 東京工科大学, mar 2021.

[9] Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[10] Ran Jing. A self-attention based lstm network for text classification. In *Journal of Physics: Conference Series*, Vol. 1207, p. 012008. IOP Publishing, 2019.

[11] Li Lei, Ming Chen, Chengwan He, and Duojiao Li. Xss detection technology based on lstm-attention. In *2020 5th International Conference on Control, Robotics and Cybernetics (CRC)*, pp. 175–180. IEEE, 2020.

[12] Cwe-79. `https://jvndb.jvn.jp/ja/cwe/CWE-79.html`. (Accessed on 10/10/2021).

[13] A hybrid of cnn and lstm methods for securing web application against cross-site scripting attack — waheed kadhim — indonesian journal of electrical engineering and computer science. `http://ijeecs.iaescore.com/index.php/IJEECS/article/view/23131`. (Accessed on 10/10/2021).

[14] Alex Graves, Santiago Fernández, and Jürgen Schmid-

huber. Bidirectional LSTM networks for improved phoneme classification and recognition. In Wlodzislaw Duch, Janusz Kacprzyk, Erkki Oja, and Slawomir Zadrozny, editors, *Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005, 15th International Conference, Warsaw, Poland, September 11-15, 2005, Proceedings, Part II*, Vol. 3697 of *Lecture Notes in Computer Science*, pp. 799–804. Springer, 2005.

[15] Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron C. Courville. Ordered neurons: Integrating tree structures into recurrent neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[16] Beautiful soup: We called him tortoise because he taught us. `https://www.crummy.com/software/BeautifulSoup/`. (Accessed on 10/10/2021).

[17] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, Vol. 1, pp. 108–116, 2018.

[18] publictrain/pekopeko_soup: peko-n. `https://github.com/publictrain/pekopeko_soup`. (Accessed on 10/10/2021).