Vol.2021-DPS-186 No.20
Vol.2021-CSEC-92 No.20
2021/3/15

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

飯野 和真[1,a)]　宇田 隆哉[1,b)]

概要：機械学習を用いてXSS攻撃を検出する研究が行われているが，実際の攻撃データセットを使用していると主張しながら実質無害なコードを使用していたり，コードの内容が不鮮明であったり，機械学習の前処理が不適切であったりするものがある．このようなデータセットや処理方法を使用して機械学習の検出精度を論じても，実際の攻撃を同等の精度で検出することはできない．本論文では，これらの研究がなぜ不適切であるのか明らかにする．一方で，評価に使用する攻撃データをどのように用意し，どのように前処理を行えば適切となるのかについて論じる．さらに，不適切なデータセットまたは前処理と，我々が用意した適切なデータセットを適切に前処理した場合とで，精度にどれだけの差が出るのか評価した．結果として，機械学習のアルゴリズムによっては，F値に0.9以上の差が出るものがあることが判明した．本研究は，機械学習を用いてXSS攻撃検知を行う研究者が注意すべき点について論じ，また数値で評価しており，有用である．

Explanation of XSS Attack Detection Researches by Machine Learning with Inappropriate Dataset or Processing and Comparison of Accuracy

1. はじめに

Webアプリケーションの脆弱性を利用した攻撃に，クロスサイトスクリプティング（以下，XSS）攻撃があり，攻撃によって，クッキー値の漏洩や，サイトの改変，悪質なサイトへの誘導などの被害が生じる可能性がある．既存研究に，機械学習アルゴリズムを用いてXSS攻撃を検出するものがあるが，実際の攻撃データセットを使用していると主張しながら実質無害なコードを使用していたり，コードの内容が不鮮明であったり，機械学習の前処理が不適切であったりするものがある．このようなデータセットや処理方法を使用して機械学習の検出精度を論じても，実際の攻撃を同等の精度で検出することはできない．

本論文では，既存研究のデータセットのどこに問題があるのか，どうしてそれが問題となるのかについて明らかにする．また，評価に使用する攻撃データを適切な形で処理するにはどのようにすればよいのかについて論じる．我々は，適切な意味での実際のXSS攻撃データセットを探したが，公開されているものを見つけることはできなかった．そこで，既存研究で使用されている攻撃データセットから，難読化ツールを使用して実際の攻撃に準ずる形式のデータセットを作成し，機械学習による評価を行った．なお，2種類の難読化ツールを用いて，未知のデータに対する精度が適切に評価されるようにしている．

最終的に，既存研究の不適切な方法と，我々が作成した実際の攻撃に準ずる形式のデータセットを用いた方法の両方で機械学習を行い，その精度を比較した．

[1] 東京工科大学
東京都八王子市片倉町 1404-1, 192–0982
[a)] c0117021f2@edu.teu.ac.jp
[b)] uda@stf.teu.ac.jp

2. 関連研究

2.1 データセットの不適切な使用による問題

本節では，データセットを不適切に扱った研究とその問題について説明する．具体的には，データセットそのものに問題があるものと，IPアドレスとFQDNを置換せずに扱っているものがある．前者については，2.1.1項にて理由を述べた後，続く項にて研究例を紹介する．

2.1.1 alertなどの実質無害なコードの問題点

関連研究について説明する前に，なぜalertといったコードで構成されたデータセットを，攻撃データとして扱うこ

とが不適切なのか説明する．そもそも alert の機能は，任意のテキストをポップアップ表示することである．

実質無害な XSS コードを掲載している代表例として XSSED[*1]がある．XSSED には脆弱性のあるサイトの一覧が掲載されており，ミラーサイトも掲載されている．また，アクセスする URL には script タグによって挿入された JavaScript が含まれており，実際にアクセスするとそのスクリプトが実行される．しかし，そのスクリプトはクッキー情報を得たりフィッシングサイトへリダイレクトしたりするものではなく，alert のような実質無害なスクリプトである．実際に，XSSED に掲載されているミラー[*2]に

アクセスして確認したところ，XSS と表示されるポップアップが出現した．挿入されているコードは，<script>alert('/xss/')</script> であり，ポップアップが表示されるだけである．alert 以外にも，prompt も実質無害なコードとして使用可能である．

どれだけ機械学習のアルゴリズムやパラメータを改善して分類しても，実質無害なコードでトレーニングを行っては，実際の攻撃コードに対して同様の効果を発揮できない．

2.1.2 データセットが不鮮明な研究

Liu らは，脆弱性を含むテキストを分析し脅威度の評価を行う研究を行っている [1]．深層学習を用いて，The NVD vulnerability library のデータを使用しているが，現在そのデータは一切掲載されておらず，使用したデータが適切であったのか評価することができない．Guichang らは，CNNPayl に基づく XSS 攻撃の検知を行う研究を行っている [2]．使用されたデータは，xss-sql dataset であると述べられているが，参照先はこのデータセットではなく，そのデータセットを使用している論文であった．そのため，参考文献に記述された論文を調べ，データセットの参照先を調査し URL を入手した．しかし，現在そのサイトにアクセスすることができず，使用したデータが適切であったのか評価することができない．

Choi らは，N-gram を用いて XSS の特徴を抽出し，その後 SVM で分類する研究を行っている [3]．用いられたデータセットは，Web 攻撃の専門家によって提供されたデータを使用していると述べられているのみで，内容が適切であるか評価できない．Komiya らは，SVM や NB および k 近傍法を用いて，悪意のある Web コードの分類を行っている [4]．用いられたデータセットは，専門家の協力によって収集されたと述べられているのみで，内容が適切であるか評価できない．

Banerjee らは，SVM や KNN，Random Forest やロジスティック回帰を用いて，Web アプリケーションでの XSS 検知を行っている [5]．使用した攻撃データは，Kaggle から取得したと述べられているが，Kaggle に掲載されている XSS に関連するデータセットは複数あり，どのデータセットを使用したか不明である．参考までに，Kaggle の datasets で “XSS” という単語でトップに出てきた Cross site scripting XSS dataset for Deep learning[*3]について調査した．このデータセットが XSS のみに着目した唯一のデータセットであり，JavaScript が挿入されたコードで構成されている．他のデータセットには，HTTP リクエストで構成されているものや，中身に全くデータがないものなどが見受けられた．Cross site scripting XSS dataset for Deep learning のうち，攻撃であるとラベル付けされた各スクリプトに対し，alert という単語を含むか調査した．なお，前処理として，HTML エンティティのデコードや，全角から半角への変換，全文字列の小文字化を行った．その結果，攻撃データ全体の約 98%が alert という単語を含んでいた．このデータセットは，portswigger チートシートと owasp チートシートをもとに作成されているため，このような結果になったと考えられる．alert を含まなかったスクリプトも，href 属性や src 属性の属性値が Google の URL となっていたり，prompt を使用していたりしていた．つまり，Banerjee らが使用したデータセットは，出典が不鮮明であるだけでなく，その中で最も適切と思われるデータセットが使われていたと仮定しても，攻撃コード全体の約 98%が実質無害なコードで構成されており，不適切といえる．

2.1.3 データセットに XSSED を用いている研究

Vishnu らは，機械学習を用いて XSS を検知する研究を行っている [6]．使用した機械学習アルゴリズムは，SVM や NB，決定木であり，攻撃データは XSSED から収集されている．

Habibi らは，N-gram を使用した機械学習による XSS の検出を行う研究を行っている [7]．機械学習アルゴリズムには，SVM や KNN，NB が用いられているが，使用したデータは XSSED から取得したものであった．

Akaishi らは，XSS を単語に分割し頻度と共起度を用いてベクトル化した上で，機械学習を用いて分類する研究を行っている [8]．機械学習アルゴリズムには，CNN や SVM，MNB が用いられており，前処理では，IP アドレスや FQDN の置換が適切に行われている．しかし，使用しているデータは XSSED から取得たものである．

Krishnaveni らは，悪意のある Web ページを機械学習によって予測する SpiderNet と呼ばれるツールを開発している [9]．機械学習アルゴリズムには SVM と ELM を用いているが，使用されたデータはすべて XSSED から取得されたものである．

Fang らは，深層学習による XSS 検知の研究を行ってい

[*1] http://www.xssed.com/
[*2] www.college.harvard.edu, XSS Vulnerability, http://www.xssed.com/mirror/76647/, (2021.1.25 accessed)
[*3] https://www.kaggle.com/syedsaqlainhussain/cross-site-scripting-xss-dataset-for-deeplearning

```
var keys='';
document.onkeypress = function(e) {
  get = window.event?event:e;
  key = get.keyCode?get.keyCode:get.charCode;
  key = String.fromCharCode(key);
  keys+=key;
}
window.setInterval(function(){
  new Image().src = 'http://hack.com/keylogger.php?c='+keys;
  keys = '';
}, 1000);
```

図 1　キーロガーのスクリプト例
Fig. 1　Script sample for key logging.

る [10]．前処理としてデコードを行っており，保護が必要な情報は削除されている．また，特徴抽出には word2vec を用いている．しかし，使用されているデータセットが XSSED である．

2.1.4　データセットに XSSED 以外を用いている研究

Li らは，SVM を用いて，ユーザが入力したパラメータが悪意のあるものかどうか識別する研究を行っている [11]．データセットには脆弱性診断ツールを用いて送信されるデータを使用しており，評価には Exploit-DB のデータを使用している．このツールから送信されるデータが，実際に有害か無害かは判断できないが，評価に使用される Exploit-DB では，攻撃の手順は掲載されているものの，実際に被害を与えるコードは掲載されていない．代わりに，掲載されているのは alert といった無害化されたコードである．そのため，Exploit-DB を用いるのは，実際の攻撃を検知する上で不適切である．

Rietz らは，Web アプリケーションに対する新たなファイアウォールアーキテクチャを提案している [12]．評価に使用されているデータは，脆弱性診断ツールを用いて悪性と出力された HTML である．しかし，このツールの判断基準が非公開であるため，取得された HTML が実際に害を与えるものであるか判断できない．

Kaur らは，機械学習を用いてブラインド XSS の検知を行っている [13]．データは複数のサイトから取得しており，XSS のチートシートが掲載されているサイトや，脆弱性を診断するためのコードが多数掲載されているサイト，GitHub で公開されているコードから収集している．しかし，どのスクリプトも実際の攻撃ではなく，alert といった無害なコードとなっていることを確認した．

研究として論文が存在するわけではないが，適切な攻撃コードが掲載されているサイトも存在する．例えば，XSS Payloads[*4] には，実際に被害を与えることのできるスクリプトが掲載されている．しかし，実際に攻撃に使用されたスクリプトが掲載されているのではなく，有志によって攻撃スクリプトの例が紹介されているだけである．例を図 1 に示す．

図のソースコードには “get.keyCode” や “keylogger” のような記述が含まれており，“hack.com” というドメイン名も見られる．これをこのまま機械学習に入力すると，このような記述を含むものを悪性，そうでないものを良性とするモデルが作成されてしまう．実際に攻撃を行う際に，攻撃者がこのような記述を使うとは考えにくく，そのモデルを使った評価は適切とはいえない．

2.1.5　IP アドレスや FQDN の置換を行っていない研究

IP アドレスや FQDN は可変な値であるため，機械学習に影響を及ぼす．しかし，2.1.2 項にて挙げた研究の中で，Liu ら，Choi ら，Komiya ら，Banerjee らのものは，特徴検出や分類を行う際に IP や FQDN の置換をしていなかった．一方で，Guichang らの研究では，HTTP や HTTPS のリンクの正規化を行っていた．

2.1.3 項にて挙げた研究の中では，Vishnu ら，Habibi ら，Krishnaveni らが IP アドレスや FQDN の処理を無視していた．一方，Akaishi らは，IP アドレスと FQDN の置換を行うのみならず，置換した場合とそうでない場合の分類精度を比較していた．Fang らは，URL に対して置換を行っていたが，IP アドレスに対する置換は行われていなかった．

2.1.4 項にて挙げた研究の中では，Li ら，Rietz ら，Kaur らすべてが IP アドレスや FQDN の処理を無視していた．

以上より，本章にて関連研究として挙げた全 13 論文のうち，IP アドレスや FQDN の置換や正規化などの処理を全く行っていなかった研究は 10 あり，全体の 7 割を超えている．特徴抽出手法の優劣を競ったり，機械学習アルゴリズムのチューニングにより攻撃の検出精度を上げたりすることも重要ではあるが，それ以前に使用するデータセットが実際の攻撃を検知する上で適切であるかどうか判断する必要がある．

2.2　不適切ではない類似研究とその目的

本節では，不適切ではない類似研究とその目的について説明し，本研究との違いを明らかにする．

2.2.1　Eucalyptus を用いた XSS に関する研究

蓄積型 XSS(Stored XSS) は，Web サイトが蓄積するコンテンツ中にスクリプトを挿入させておくことで，ユーザがアクセスした際に被害を起こすものである．Somorovsky らは，Amazon の制御インターフェースとプライベートクラウドソフトウェアである Eucalyptus を用いたシステムに対して，XSS 攻撃が可能かどうかの実験を行い，その XSS 攻撃が成功したと報告している [14]．具体的には，ディスカッショントピックの生成時に，任意の HTML を挿入し，Web サイトに蓄積させることで攻撃を行った．このトピックの見出しはエンコードされずに HTML に反映されるため，攻撃が成立する．挿入したスクリプトが実行されないようパディング技術が使用されていたが，パディングの位置がわかれば攻撃が可能であると述べられている．また，

[*4] http://www.xss-payloads.com/index.html (2021.1.25 accessed)

このような攻撃は，XSS を防ぐソフトウェアや組み込みメカニズムを使用していても防げず，Cookie 情報や個人情報の漏洩に繋がるとも述べられている．

Somorovsky らの研究目的は，制御インターフェースにおける脆弱性によって，XSS 攻撃が可能であることを証明することである．また，その攻撃の対策として，ブラックボックス分析手法の紹介をすることである．一方，本研究は，既存システムに対して XSS 攻撃が可能かどうか評価するのではなく，XSS 攻撃のデータセットを適切に扱い，機械学習を用いて検出を行うとどうなるか評価するものである．

2.2.2 フィルタによる XSS 防御手法

Tripp らは，XSS 攻撃を無害化する PHP サニタイザについて述べている [15]．具体的には，まず HTML タグからすべてのスペースを削除し，タグ内の属性を実行させなくしている．ただし，属性値がダブルクォーテーションで挟まれている場合は，タグ内の href 属性が保持される．次に，img タグを削除し，加えて，href 属性に含まれる JavaScript ディレクティブを削除する．最後に属性のないすべてのタグを削除する．このような処理を行うことで，良性の値を持つ <a> タグのみが許可される．このフィルタによって，例えば，<script>alert('XSS')</script> は属性のないタグが削除されたことによって失敗する．また，<script src="www.attacker.com"></script> や <input type="text" onfocus="alert('XSS')"/> のような攻撃も，スペースが削除されたことによって失敗する．ただし，スペースをタブに置き換えることでフィルタを通過できるとも述べられている．

Tripp らの研究目的は，Web アプリケーションへの脆弱性診断を従来より効果的に行うことである．そのために，Web サイトからの応答を，学習アルゴリズムを用いて分析している．一方，本研究は，Web アプリケーションの脆弱性検出ではなく，XSS 攻撃の検出を行うものである．

2.2.3 データフロー解析と機械学習を用いたソフトウェアの脆弱性検出手法

Kronjee らは，データフロー分析と機械学習を組み合わせて，SQL インジェクションと XSS 攻撃を検知する手法を提案している [16]．データセットには，NIST が提供する National Vulnerability Database のデータセットと，SAMATE データセットが使用されており，PHP コードが分類されている．具体的には，到達定義分析や到達定数分析などのデータフロー分析手法をデータセットに対して行い，特徴を抽出した後，機械学習を用いて分類を行っている．

Keonjee らの研究目的は，PHP アプリケーションに含まれる，XSS 攻撃や SQL インジェクション攻撃の脆弱性を検出することである．一方，本研究は，PHP アプリケーションの脆弱性検出ではなく，XSS 攻撃の検出を行うものである．

3. 提案手法

2 章で述べたように，使用しているデータセットに問題がある研究や前処理に問題がある研究があった．本章では，その解決策となる具体的な手法について述べる．

3.1 適切なデータと必要な前処理

データセットに使用される適切なデータである要件を次に示す．

[要件 1] alert タグを含むものをデータとして使用しない
[要件 2] 正常データと攻撃データは 10 語以内の単純なパターンマッチングによる区別ができない
[要件 3] 正常データには URL が必ず含まれる
[要件 4] IP アドレスと FQDN は置換し表現を統一する
[要件 5] トレーニングに使用するデータをテストに使用しない

[要件 1] は，2.1.1 項にて理由を述べた通りである．[要件 2] は，そもそも単純なパターンマッチングで完全な分類が行えるのであれば，機械学習を必要としないためである．[要件 3] は，攻撃データには URL が，厳密にいえば IP アドレスや FQDN が含まれるためである．URL の有無によって分類が可能であれば，機械学習をする必要がない．[要件 4] は，2.1.5 項で挙げた研究にみられたもので，データセット内の IP アドレスや FQDN に偏りがある場合，これをもとにパターンマッチングで分類できてしまう可能性があるためである．[要件 5] は，テストデータは未知でなければならないためである．

3.2 データ収集

[要件 3] を満たすため，正常データは JavaScript 内に URL を含み，他のサイトへアクセスさせるようなものとする．正常データを取得する際には，HTTrack Website Copier[*5] を使用し，.html ファイルや.js ファイルを集め，JavaScript で記述されている部分を抽出した．また，[要件 1] を満たすため，攻撃データとして実際に被害を与えるコードを探したが，実際に使用された攻撃コードで公開されているものを見つけることはできなかった．なお，XSS Payloads に実際に被害を与えることのできるスクリプトが掲載されている．ただし，それらは例であり，39 個しか存在しない．そこで，本研究では，正常データとこのデータを組み合わせ，攻撃データを生成した．具体的な手法は次節にて述べる．

3.3 前処理

前処理として，前節で述べた 39 個の攻撃コードを，難

[*5] https://www.httrack.com/

読化ツールを用いて難読化した．図 1 に示したとおり，攻撃者が同じ攻撃を行う際には，同様のコードを記述する必要がある．しかし，そのコードが攻撃コードであることが容易に察知されないように，余分なコードを追加してカモフラージュするであろうし，変数名も攻撃とは関係しないものとするであろう．我々は，この作業が，難読化ツールが行うものと同様であると考えた．もちろん，難読化ツールが使用する変数名は，攻撃者が想定するような変数名とは異なるであろうが，ベクトル化されて機械学習のネットワークに入力されてしまえば，その違いはなくなると考えた．

なお，[要件 5] を満たすため，収集した 39 個のコードを，2 つの難読化ツールを用いて，2 種類のデータとして生成した．攻撃者が同じ攻撃を行う際には，類似のコードを記述する必要があるが，同一の難読化ツールで難読化したものをトレーニングとテストの両方に入力すると，その特徴が同一になってしまうためである．厳密には，異なる難読化ツールを使用したとしても，1 つの攻撃コードから派生して 2 種類のデータを作成し，片方をトレーニングに，もう片方をテストに入力すると，それは未知のデータテストにはならない．しかし，ここで挙げられているものは，基本的に必要最小限の記述で書かれた攻撃コードの例であり，同様の攻撃をするのであれば，この記述を含む必要がある．つまり，必要最小限の記述で書かれた攻撃コードを真似て，2 人の攻撃者が攻撃コードを個々に作成し，その片方が先に防御側に回収されて，防御フィルタの作成に使われていると考えれば，2 つの難読化ツールが模擬的にそれぞれの攻撃者に該当すると考えることができる．本提案手法は，この考えに基づいている．本論文では，攻撃者が攻撃コードを作成する際，元のコードを同様に改変して偽装すると考え，同様の処理を難読化ツールを使って行っている．

[要件 5] に関しては，本来，1 つのコードから派生した 2 種類のコードを，トレーニングとテストのそれぞれに入力することは禁止されているが，行いたい攻撃内容を最もシンプルな形で書いたものが収集した 39 個のコードとすると，それを様々な攻撃者が参照して偽装するとすると，事実上 [要件 5] を満たすと判断した．

次に，正常データを用いた攻撃データの生成について述べる．正常データは，JavaScript のコード部分のみでありHTML を含まない．ただし，JavaScript 内に HTML を出力するコードなどがある場合，その内容は含むものとする．手法としては，難読化された攻撃コードを複数に分割し，正常コードにランダムに挟み込むことで攻撃データを生成する．具体的には，難読化したコード，正常コード共に，セミコロンの位置で複数に分割する．

攻撃コード自体が壊れないよう，後から挿入される攻撃コードのパーツが，既に挿入されているパーツよりも前に挿入されないようにする．また，挿入箇所がなくなった場合，正常コードの最後に残りをすべてを追加する．これらの処理を，2 種類の難読化ツールを用いて，同一の攻撃コードと正常コードの組み合わせに対して必要な数だけ行う．なお，難読化処理には Javascript Obfuscator[*6] と JavaScript Obfuscator Tool[*7] を使用した．

表 1　word2vec で使用するデリミタ
Table 1　Delimiters for word2vec.

.	,	;	:	!	%	<	>	@	~
&	*	+	=	-	$	’	?	[	]
(	)	#	{	}	”	_	|	/	\

次に，[要件 2] を満たすため，すべての正常データと攻撃データに対して，同一ツールで最適化処理を行う．難読化されたコードの変数名などには特徴があり，これをもって正常データと区別できてしまうことは [要件 2] に反する．最適化処理を行えば，変数名や関数名は最も短い表記に変更され，改行や無意味な空白も削除されるので，前述の特徴は消える．また，難読化ツールによって変数名や改行位置に特徴がある場合もあり，これをトレーニング時に強い特徴とされてしまうと，別の難読化ツールを使ったテストデータに対して精度が下がってしまう．この最適化処理はそれも避けることができる．なお，最適化処理には YUI Compressor[*8] を使用した．

また，IP アドレスと FQDN の扱いについて説明する．すべての攻撃データ，正常データの IP アドレスと FQDN を，それぞれ ‘IPaddress’ と ‘FQDN’ に置換した．

次に，word2vec によるベクトル化する際の前処理について説明する．まず，全角文字を半角に，アルファベットをすべて小文字に変換する．加えて，HTML エンティティのデコードも行う．そして，単語に分割する処理を行う．分割に使用するデリミタを表 1 に示す．なお，特殊文字が連続する場合や，最初および最後の文字が特殊文字であった場合に，中身のない要素ができてしまう．そのため，中身のない要素を削除する．

word2vec によるベクトル化については次節にて説明する．

正常データと攻撃データを用いて評価を行う際のデータ分割方法について図 2 に示す．

図に示すように，5 グループに分割したデータの 1 グループをテストに，残りをトレーニングに使用する．なお，攻撃データ A の 1 グループでテストを行う場合，同一の攻撃コードを使って同じアルゴリズムで生成されたサンプルがトレーニングとテストの両方に含まれるため，事実上の 5 分割交差検証となる．攻撃データ B の 1 グループでテストを行う場合，同一の攻撃コードを参照していても，難読化

*6 https://javascriptobfuscator.com/Javascript-Obfuscator.aspx
*7 https://obfuscator.io/
*8 http://yui.github.io/yuicompressor/

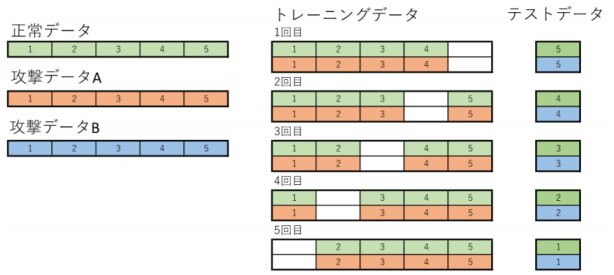

図 2　テスト時のデータ分割手法
Fig. 2　Data division method for tests.

時のアルゴリズムがトレーニングとテストで異なるため，こちらはテストとなる．

3.4　word2vec によるベクトル化と分類

ベクトル化をする際に，テストデータの知識がトレーニングデータに反映されることは不適切である．そこで，全データを使ってベクトルを作成してから 5 分割を行うのではなく，3.3 節で述べた分割を行った後，トレーニングデータとなったデータの攻撃データのみを用いて，ベクトル化のモデルを作成する．そして，そのモデルを用いて，3.3 節の方法で単語に分割されたトレーニングデータ，テストデータをベクトル化する．

モデルの作成に全データを用いていないため，モデルに存在しない単語があり，ベクトル化できない場合があるが，その単語はすべてを 0 の要素とした配列とする．

word2vec でトレーニングする際には，Akaishi らの研究を参考にし，gensim ライブラリの Word2Vec 関数を使用する．word2vec の出力次元数は特に決まりはないが，実験環境のパフォーマンスを考慮して，本研究では 10 としている．

ベクトル化されたトレーニングデータおよびテストデータは，もともとの単語数が異なるためそれぞれサイズが異なる．そこで，最も大きいサイズに合わせて，すべてを 0 の要素とした配列を追記してパディングする．

4.　評価

3.1 節で述べた要件を考慮し，評価を行った．

まず，[要件 1] を満たさないものの例として，Kaur らが用いていたチートシート[*9]のデータを攻撃データとしたものを用意した．このチートシートは，alert といった実質無害なコードで構成されている．また，機械学習を用いなくても，if 文で容易に分類できるか実験を行った．

また，[要件 4] については，IP アドレスと FQDN を置換しなかった場合についての評価を行った．

さらに，[要件 5] については，トレーニングデータ作成に使用した難読化ツールと同一のものがテストデータに使われている場合と，異なるものがテストデータに使われている場合との比較を行った．

[*9] Top 500 Most Important XSS Cheat Sheet for Web Application Pentesting, https://gbhackers.com/top-500-important-xss-cheat-sheet/

表 2　要件を満たした場合の評価
Table 2　Evaluation with correct data.

アルゴリズム	正解率	適合率	再現率	F 値
CNN	0.600	0.556	0.999	0.714
	(0.026)	(0.017)	(0.002)	(0.013)
SVM	0.531	1.000	0.061	0.114
	(0.015)	(0.000)	(0.030)	(0.052)
RF	0.527	0.987	0.055	0.103
	(0.010)	(0.027)	(0.021)	(0.038)

表 3　チートシートを攻撃データとした場合の評価
Table 3　Evaluation with attack data from cheat sheet.

アルゴリズム	正解率	適合率	再現率	F 値
CNN	0.970	0.966	0.975	0.970
	(0.024)	(0.027)	(0.022)	(0.024)
SVM	0.915	0.861	0.993	0.922
	(0.022)	(0.036)	(0.010)	(0.019)
RF	0.984	0.995	0.973	0.984
	(0.008)	(0.006)	(0.015)	(0.009)

機械学習のアルゴリズムについては，CNN，SVM，Random Forest (RF) を用いた．実装には keras2.4.3 ライブラリと scikit-learn 0.23.1，numpy 1.18.5 を使用した．CNN のネットワーク構成，カーネルサイズ，プーリングサイズ，活性化関数は落合らの研究 [17] に合わせた．

SVM は svm ライブラリの SVR 関数と fit 関数を用いて実装を行い，カーネルは RBF とした．RF は RandomForestClassifier ライブラリの RandomForestClassifier 関数と fit 関数を用いて実装した．

4.1　要件を満たした場合の評価

まず，3.1 節で挙げた 5 つの要件すべてを満たした場合について評価を行った．結果を表 2 に示す．表の数値は平均値であり，括弧内の数値が標準偏差である．攻撃データ数は 400，正常データ数も 400 とした．

4.2　チートシートを使用した評価

攻撃データに，Kaur らが用いていたチートシートを使用した時の分類結果を表 3 に示す．表の数値は平均値であり，括弧内の数値が標準偏差である．攻撃データ数は 400，正常データ数も 400 とした．

また，チートシートには ‘alert’ が頻発するといった特徴がみられるため，if 文を用いてどの程度分類できるか実験を行った．条件はコード内に ‘alert’ という文字列を含むかどうかである．なお，事前に，HTML エンティティのデコードや，全角から半角の変換，及び小文字への変換といった前処理を行っている．攻撃データ数は 400，正常データ

表 4　チートシートの攻撃データに対する条件文による分類
Table 4　Classification of attack data from cheat sheet by conditional statements.

アルゴリズム	正解率	適合率	再現率	F 値
if 文	0.871	0.990	0.750	0.853

表 5　IP アドレスと FQDN を置換しなかった場合の評価
Table 5　Evaluation with actual IP address and FQDN.

アルゴリズム	正解率	適合率	再現率	F 値
CNN	0.598	0.555	0.999	0.713
	(0.018)	(0.012)	(0.002)	(0.010)
SVM	0.532	1.000	0.063	0.118
	(0.011)	(0.000)	(0.023)	(0.040)
RF	0.545	0.978	0.092	0.166
	(0.020)	(0.044)	(0.040)	(0.068)

表 6　テストデータに同一の難読化ツールを使った場合の評価
Table 6　Evaluation with the same obfuscator for test data.

アルゴリズム	正解率	適合率	再現率	F 値
CNN	0.997	0.995	0.999	0.997
	(0.002)	(0.003)	(0.002)	(0.002)
SVM	0.974	1.000	0.947	0.973
	(0.007)	(0.000)	(0.014)	(0.007)
RF	0.995	1.000	0.989	0.994
	(0.004)	(0.000)	(0.009)	(0.004)

数も 400 とし，5 分割せずに評価した．結果を表 4 に示す．

4.3　IP アドレスと FQDN の置換に関する評価

IP アドレスと FQDN を置換しないでそのまま扱った場合，分類結果がどうなるか評価した．結果を表 5 に示す．表の数値は平均値であり，括弧内の数値が標準偏差である．攻撃データ数は 400，正常データ数も 400 とした．

4.4　難読化ツールを固定した場合の評価

トレーニングデータとテストデータが同じアルゴリズムで作成されている場合，テストデータの知識がトレーニングに影響を与えるため，テストとはならずに検証となる．本論文では，3.3 節で述べたとおり，異なる難読化による変換を異なる攻撃者によるコードの作成と同等とみなし，同一の難読化ツールでトレーニングデータとテストデータを作成した場合の評価を行った．結果を表 6 に示す．表の数値は平均値であり，括弧内の数値が標準偏差である．攻撃データ数は 400，正常データ数も 400 とした．

5.　考察

表 2 より，要件をすべて満たした場合の評価は，最も F 値の高い CNN で 0.714，最も低い RF で 0.103 となった．SVM は 0.114 であるため RF の結果に近い．本節では，この値を基準として，不適切な方法で評価を行った場合の結果について考察する．

5.1　チートシートを使用した評価の考察

表 3 より，チートシートを攻撃データに使用した場合，CNN，SVM，RF のいずれも要件をすべて満たした場合の評価結果よりも F 値が高くなっていることがわかる．とくに，SVM と RF は 0.9 以上も F 値が上昇している．表 4 より，このチートシートの攻撃データは，‘alert’ を if 文で処理しただけで 0.853 もの F 値となっている．つまり，‘alert’ を含んだものを攻撃データとして評価を行うことは不適切であり，if 文だけで分類可能なものを機械学習で評価するまでもないことが明確となった．

5.2　IP アドレスと FQDN の置換に関する評価の考察

IP アドレスと FQDN を置換しないでそのまま扱った場合の分類結果は，表 5 より，CNN，SVM，RF のいずれの F 値も，要件をすべて満たした場合の評価と比較して 0.1 も違いがないことが分かった．一方，Akaishi らの研究では，FQDN と IP アドレスを置換しなかった場合の正解率が，した場合の正解率より明確に高くなっていた．Akaishi らは，正常データを大手サイトから集めており，そのせいでスクリプトに含まれる IP アドレスや FQDN が絞られてしまい，分類が容易になった可能性がある．また，Akaishi らのデータは [要件 5] を満たしていないため，その影響も考えられる．

5.3　難読化ツールを固定した場合の評価に関する考察

表 6 より，同一の難読化ツールをトレーニングデータとテストデータの両方に使用した場合，CNN，SVM，RF のいずれも要件をすべて満たした場合の評価結果よりも F 値が高くなっていることがわかる．とくに，SVM と RF は 0.9 以上も F 値が上昇している．最適化ツールを使用しているため，変数名，関数名，無意味な空白，改行は分類における特徴とはならない．つまり，同一目的の攻撃を行う際の，コードのカモフラージュの特徴が分類結果に大きな影響を与えていることになる．実際に攻撃に使用されたコードが入手できなかったため，今回は擬似的な評価となったが，同一攻撃者が書いたコードをトレーニングできているかどうかが，テストの結果に大きく左右すると想定できる．同時に，同一攻撃者が書いたコードをトレーニングできていることを前提としたアルゴリズムの評価は，不適切であることが明確となった．

6.　結論

本研究では，機械学習を用いて XSS 攻撃を分類する研究に着目し，関連研究における実験が適切なものであるかどうか評価した．結論として，関連研究で適切な攻撃データを適切な形で扱っていると断言できるものはなかった．な

お，出所が不鮮明なデータセットを使用していて内容を確認できない研究もあるため，すべてが不適切であるとはいえない．具体的には，実際の攻撃に使用されない ‘alert’ を含んだものを攻撃データとしているものや，IP アドレスと FQDN を置換せずにそのまま用いているものが存在した．

実際に被害を与えた XSS のデータセットを我々は見つけることができなかったため，本論文で適切なデータセットの要件を定義し，その要件をすべて満たすデータセットを擬似的に作成して評価した．結果の F 値は，CNN が 0.714，SVM が 0.114，RF が 0.103 であった．一方，‘alert’ を含むデータセットでは，これより高い値で分類されることがわかり，if 文だけでも 0.853 の F 値となることがわかった．IP アドレスと FQDN を置換しないデータセットに関しては，要件をすべて満たしたものの F 値と比較して 0.1 も差がなかったが，これはデータセット中の IP アドレスと FQDN に偏りがないためではないかと推測された．

さらに，同一の難読化ツールをトレーニングデータとテストデータの両方に使用した場合についても評価を行った．結果の F 値は，CNN，SVM，RF のいずれも要件をすべて満たしたデータセットのものより高い値となった．

本論文により，不適切な関連研究の分類精度が，どの程度信用できないものか明らかになった．なお，要件をすべて満たしたデータセットにおける評価の F 値は，チューニングしていない CNN で 0.714 であったため，XSS に特化したチューニングを行うことでどこまで改善されるのか，今後研究していきたい．

謝辞　本研究は JSPS 科研費 JP18K11248 の助成を受けたものです．

参考文献

[1] Liu, K., Zhou, Y., Wang, Q. and Zhu, X.: Vulnerability Severity Prediction With Deep Neural Network, In *Proc. 5th International Conference on Big Data and Information Analytics (BigDIA)*, pp.114–119, (2019).

[2] Guichang, Z., Yunfeng, N. and Xing, W.: CNNPayl: An Intrusion Detection System of Cross-site Script Detection, In *Proc. 11th International Conference on Machine Learning and Computing (ICMLC)*, pp.477–483, (2019).

[3] Choi, J., Kim, H., Choi, C. and Kim, P.: Efficient Malicious Code Detection Using N-Gram Analysis and SVM, In *Proc. 14th International Conference on Network-Based Information Systems*, pp.618–621, (2011).

[4] Komiya, R., Paik, I. and Hisada, M.: Classification of Malicious Web Code by Machine Learning, In *Proc. 3rd International Conference on Awareness Science and Technology (iCAST)*, (2011).

[5] Banerjee, R., Baksi, A., Singh, N. and Bishnu, S. K.: Detection of XSS in Web Applications Using Machine Learning Classifiers, In *Proc. 4th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech)*, (2020).

[6] Vishnu, B. A. and Jevitha, K. P.: Prediction of Cross-Site Scripting Attack Using Machine Learning Algorithms, In *Proc. International Conference on Interdisciplinary Advances in Applied Computing (ICONIAAC)*, Article No.55, pp.1–5, (2014).

[7] Habibi, G. and Surantha, N.: XSS Attack Detection With Machine Learning and n-Gram Methods, In *Proc. International Conference on Information Management and Technology (ICIMTech)*, pp.516–520, (2020).

[8] Akaishi, S. and Uda, R.: Classification of XSS Attacks by Machine Learningwith Frequency of Appearance and Co-occurrence, In *Proc. 53th Annual Conference on Information Sciences and Systems (CISS)*, (2019).

[9] Krishnaveni, S. and Sathiyakumari, K.: SpiderNet: An Interaction Tool for Predicting Malicious Web Pages, In *Proc. International Conference on Information Communication and Embedded Systems (ICICES2014)*, (2014).

[10] Fang, Y., Li, Y., Liu, L. and Huang, C.: DeepXSS: Cross Site Scripting Detection Based on Deep Learning, In *Proc. International Conference on Computing and Artificial Intelligence (ICCAI)*, pp.47–51, (2018).

[11] Li, L. and Wei, L.: Automatic XSS Detection and Automatic Anti-Anti-Virus Payload Generation, In *Proc. International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp.71–76, (2019).

[12] Rietz, R., König, H., Ullrich, S. and Stritter, B.: Firewalls for the Web 2.0, In *Proc. IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pp.242–253, (2016).

[13] Kaur, G., Malik, Y., Samuel, H. and Jaafar, F.: Detecting Blind Cross-Site Scripting Attacks Using Machine Learning, In *Proc. International Conference on Signal Processing and Machine Learning (SPML)*, pp.22–25, (2018).

[14] Somorovsky, J., Heiderich, M., Jensen, M., Schwenk, J., Gruschka, N. and Iacono, L. L.: All Your Clouds are Belong to Us: Security Analysis of Cloud Management Interfaces, In *Proc. 3rd ACM Workshop on Cloud Computing Security Workshop (CCSW)*, pp.3–14, (2011).

[15] Tripp, O., Weisman, O. and Guy, L.: Finding Your Way in the Testing Jungle: A Learning Approach to Web Security Testing, In *Proc. International Symposium on Software Testing and Analysis (ISSTA)*, pp.347–357, (2013).

[16] Kronjee, J., Hommersom, A. and Vranken, H.: Discovering Software Vulnerabilities Using Data-Flow Analysis and Machine Learning, In *Proc. 13th International Conference on Availability, Reliability and Security (ARES)*, Article No.6, pp.1-10, (2018).

[17] 落合桂一，ファティナプテリ，深澤佑介：深層学習を用いた位置依存性の高いモバイルアプリ抽出手法，情報処理学会研究報告, Vol.2018-DPS-175, No.9，pp.1–6, (2018).