ディープニューラルネットワークによる Web 攻撃検知手法の提案

藤田　良介 *　小高　知宏 **　黒岩　丈介 **　諏訪　いずみ *** 白井　治彦 ****

Proposal of Web attack detection method by deep neural network

Ryosuke FUJITA*, Tomohiro ODAKA, Jyosuke KUROIWA**
Izumi SUWA*** and Haruhiko SHIRAI****

(Received January 27, 2022)

In this paper, we proposed two feature extraction methods to detect attacks on the Web. We confirmed whether each method was effective by using deep learning.

Attacks on the Web are difficult to classify as normal communications and difficult to detect by machines. Therefore, We focused on the fact that many special symbols appear in HTTP requests for attacks on the Web.

As a result of classification using a method characterized by the number of occurrences of special symbols, the accuracy rate was about 94%.

Key words **:** Security, HTTP Request, Special Symbols, Deep Learning

1. 緒　言

近年, インターネットは急速に発展, 普及し, 今では個人がインターネットに繋がる端末を所持するようになった. それに伴って, Web もまた発展してきている. 現在では情報収集を行うための Web サイトだけでなく, 動画を閲覧する, サービスに入会, 退会する登録を行う, 物を購入する, レビューを書くといった事が行える Web アプリケーションを利用する人も増えている. また, Web アプリケーションサービスは娯楽や利便性向上を目的とするものだけではなく, 企業が業務システムに利用することも多くなっている.

Web サービスは便利であり, 多くの利用者がいる反面, Web サービスの管理には高いセキュリティが必要となっている. サービスは常に稼働し続ける必要があるものも多く, サービス利用者の名前や連絡先などの個人情報や, 企業の業務システムであれば, 社外秘となる重要情報など, 漏洩してはならない重大な情報を大量に扱う. しかし, セキュリティホールとなる脆弱性は日々, 発見され, それを狙った攻撃も度々報告されている.[1] 最近でも, Web サーバ上の任意のファイルを不正に閲覧するディレクトリ・トラバーサル攻撃にまつわる脆弱性が報告されている.[2] 他にも SQL インジェクション攻撃を始めとした情報を盗み出す攻撃や, XSS(クロスサイトスクリプティング) 攻撃といった利用者に害を与える攻撃が存在する. これらの攻撃は現在深刻な問題となっている. そのため, Web サーバ上のデータベースに格納されている重大な情報や, Web そのものを守るためにセキュリティを徹底する必要がある.

Web サービスの管理者は脆弱性を作り込まないこと, 脆弱性が発見された場合には早期に修正するといった実装面での対策の他にも攻撃から Web アプリケーション

*大学院工学研究科知識社会基礎工学専攻
**知能システム工学講
*** 仁愛女子短期大学
**** 工学部技術
*Fundamental Engineering for Knowledge-Based Society, Graduate School of Engine
**Department of Human and Artificial Intelligent Systems
***Jin-ai Women's College
****Technical Division

を保護する運用面での対策として WAF(Web Application Firewall) が存在する. WAF は Web アプリケーションを含む Web サイトと利用者間で交わされる HTTP(あるいは HTTPS) 通信を検査し, 攻撃などの不正通信を自動で遮断することができる.[3]

しかし, 現在の WAF はシグネチャ検知という手法を用いている場合が多い。シグネチャ検知はあらかじめ攻撃を識別するためのルールを記述しておき, そのルールとパターン照合を行うことによって攻撃を検知している. ただし, 現状シグネチャ検知では攻撃に用いられる入力と正常な入力を判別することが難しいという欠点がある. 特に SQL インジェクション攻撃や OS コマンドインジェクション攻撃はシグネチャ検知では回避されやすいという問題がある.

本研究では, 様々な Web への攻撃を網羅的に検知する事を目標とする. そこで, 本研究では深層学習を利用する.Web に対する攻撃には HTTP リクエスト系列に特殊記号が多く現れると考えられる. よって, 特殊記号に着目した特徴ベクトルの生成手法による検知を目指す.

本論文では 2 章で Web における攻撃の対策, および, 攻撃と特殊記号の関係を示し,3 章では提案する特徴抽出方法を示す.4 章では実験方法について述べ,5 章では実験結果を示す.6 章では得られた実験結果についての考察を述べ,7 章では本実験についてまとめる.

2. Web における攻撃と特殊記号の関係

本研究における Web は Web アプリケーションを含む Web サイトを指す. Web アプリケーションとはブラウザから様々な機能を利用可能なアプリケーションサービスの事である. クライアントとサーバ間で HTTP 通信を利用してデータの送受信を行っている. 本章では Web に対する攻撃と現在の対策方法について述べ、攻撃と HTTP リクエスト系列に含まれる特殊記号の関係を示す.

2.1 Web における攻撃と対策

本節では, 主な Web への攻撃種類と現在における攻撃への対策を述べる.

以下に主な Web への攻撃種類を示す.

- XPath インジェクション
- OS コマンドインジェクション
- Ldap インジェクション
- SSI インジェクション
- SQL インジェクション
- ディレクトリ・トラバーサル
- クロスサイト・スクリプティング

インジェクション攻撃は脆弱性を利用して情報を搾取, マルウェアの感染, データベースの改ざんや消去などを行う攻撃である. 例えば,OS コマンドインジェクションでは脆弱性のある Web アプリケーションに対して OS コマンドを送信する事で不正にアクセスする攻撃である. 攻撃を受けると, 情報を搾取される, 他の標的に攻撃する際の踏み台とされる.SQL インジェクションでは外部から不正な SQL 文を送信することで, 不正にデータベースの情報を搾取する. クロスサイト・スクリプティングでは脆弱性のある Web サイトに罠を仕掛け, サイトを訪れる利用者を悪質なサイトに誘導する事で, 利用者の情報を搾取, あるいはマルウェアの感染などを行う.

これらのインジェクション攻撃は OWASP Top 10 2021[4] では第 3 位のリスクとされている.2017 年以前の OWASP Top 10 ではインジェクション攻撃は第 1 位のリスクであった. このことからインジェクション攻撃は昔から脅威であったことが分かる.

これらの攻撃に対処するためにはまず, 脆弱性を作り込まない事が大切である. 例えば, OS コマンドインジェクションを防ぐためには, シェルを起動可能な言語機能を避ける, そもそも外部から入力された命令文をそのままコマンドラインに注入しないといった事が大切である. SQL インジェクションを防ぐためには不正な SQL 文を直接反映しないようにする必要がある. クロスサイト・スクリプティングを防ぐためには「<」や「”」といった区切りやタグの意味を持つ文字を意味の無い文字列に変換するサニタイジングが有効である.

また, セキュリティに関する情報を収集し続ける必要もある. 脆弱性は日々発見されるために, いち早く修正する必要があるからだ.

ただし, どんな人間でも, どれほど時間をかけたとしても脆弱性を全く持たない Web アプリケーションを作ることはできない. なので, 攻撃を防ぐために WAF の導入も推奨されている.

WAF はクライアントと Web サーバの間で通信を監視し, 不正アクセスを防止する. WAF の多くは検知方式にシグネチャ検知を用いている. シグネチャとはパターンを定義したものであり, WAF は Web アプリケーションへのアクセスパターンを照合し, 正常か異常かを判別している. シグネチャのパターンには主にブラックリスト型とホワイトリスト型が存在する. ブラックリスト型では既知の攻撃パターンをシグネチャに定義し, それと一致した通信パターンを拒否する. ホワイトリスト型で

表 1 各攻撃手法に出現する特殊記号 [5]

攻撃名	特殊記号
XPath インジェクション	’ = + /
OS コマンドインジェクション	; / . -
Ldap インジェクション	( ) = *
SSI インジェクション	! # - ”
SQL インジェクション	’ = +
ディレクトリ・トラバーサル	/ . \
クロスサイト・スクリプティング	< > = . ;

は正常な通信のパターンをシグネチャに定義し, それと一致しない通信のパターンを拒否する.

しかし, ブラックリスト方式ではシグネチャに定義されていない攻撃を防ぐことはできない. つまり, 未知の攻撃やまだアップデートが完了していない最新の攻撃に対応することはできない. ホワイトリスト方式ではそもそも正常な通信を定義することが難しく, 正常な通信であっても拒否する誤検知が起こってしまう可能性がある.

上記の対策は行うべきことではあるが, 現状では全て行えたとしても完璧な対策にはならず, また, Web が入力値をどう扱うのかはそれぞれのシステムによって異なるので対策も多岐に渡る. そのため, 対策漏れはどうしても起こってしまう.

2.2 Web における攻撃と特殊記号

本研究では Web における攻撃と特殊記号には強い関係があると予想し,HTTP リクエスト系列に含まれる特殊記号に着目したアプローチを行う. このアプローチ方法が有効であることは清水らの研究から証明されている.[5]

Web における各攻撃と出現頻度の多い特殊記号の関係を表 1 に示す. 例えば, OS コマンドインジェクション攻撃であれば, 入力値にシェルや命令文を入力する必要がある. これは他の攻撃も同様である. SQL インジェクション攻撃であれば, SQL 文を入力する必要があり, クロスサイト・スクリプティングであれば, JavaScript や HTML タグによる不正なスクリプトの挿入が必要とされる. 故に, 通常の入力に比べ, 特殊記号の出現頻度が大きくなると考えられる.

特殊記号に着目したアプローチ方法であれば, ある程度網羅的に攻撃を検知する事が可能であると考える. また, 未知の攻撃や WAF を回避する攻撃などにも対応可能であると期待できる.

表 2 特殊記号

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

3. 提案する検知手法

本章では侵入検知の概要と,2.2 節で示した攻撃と特殊記号の関係性から提案する侵入検知手法について述べる.

3.1 侵入検知の概要

初めに, 通常利用の通信である正常な HTTP リクエストと攻撃用の通信である異常な HTTP リクエストを用意し, 学習モデルを構築する. 本研究では 2.2 節で示した通り, Web における各種攻撃には特殊記号と関係があると予測している. それらの特殊記号に着目した特徴ベクトルを正常な HTTP リクエストと異常な HTTP リクエストから抽出する. 抽出した特徴ベクトルを用いて深層学習を行う. 特徴ベクトルの作成手法は次節で述べ, 深層学習に用いるディープニューラルネットワークは 4 章で述べる.

学習モデルによってディープニューラルネットワークを十分に学習し終えた後は, 実際にネットワークの精度を確認する. 学習モデルを作成したときと同じように検証モデルを作成する. 学習を終えたネットワークに検証モデルを入力し, その結果と検証モデルのラベルを比較することで精度を検証する. 学習モデルによって, 十分に正常なリクエストと異常なリクエストの特徴を学習することができていれば, ネットワークは正常なリクエストを正常と, 異常なリクエストを異常と分類できるはずである.

3.2 提案する特徴抽出方法

表 2 に特殊記号を示す. これらの出現頻度に着目した特徴ベクトルの抽出方法について述べる.

提案する特徴抽出方法は 2 つあり,1 つは特殊記号の出現頻度のみに着目した手法 1, もう 1 つは特殊記号だけでなく英数字の出現頻度にも着目した手法 2 である. また, 図 1 のリクエストを例として提案方法を説明する.

3.2.1 手法 1

表 3 に手法 1 で抽出する特徴を示す.

抽出する特徴の要素の 1 つ目は入力項目数がいくつ

表 3 特徴抽出方法 (手法 1)

特徴
入力項目数
各特殊記号の出現頻度
%エンコーディング
以外の特殊記号 (+ - .)

表 4 手法 1 による特徴ベクトルの例

x1	⋯	x9	x10	x11	⋯	x20	⋯	x35	⋯
		’	(	)		=		+	
2		4	1	1		1		24	

あるかである (x_1). 図 1 の例では”2”が特徴量として抽出される.

2 つ目は表 2 に示されている各特殊記号の出現頻度である ($x_2 \sim x_{34}$). 例の URL 部を除いた部分では各特殊記号は%エンコーディングが施されている. つまり, 特殊記号は”% xx”と表現されている. 例を見ると,”%27” と表現される ”””, ”%28” と表現される ”(”, ”%29” と表現される ”)”, ”%3D” と表現される ”=”が出現している. これらの出現頻度を特徴量として抽出し, 他の出現していない特殊記号は”0”として抽出する.

3 つ目は入力項目を表す”＝”や”&”以外の%エンコーディングされていない特殊記号である ($x_{35} \sim x_{37}$). 厳密には”+”などは空白を意味するなど特殊記号ではないが, 攻撃には特に”+”が多く含まれているように感じるため, これらも特徴量として含める. 例では”+”は”24”個出現しており,”.”が 1 個出現している. 他の出現していない記号は”0”として抽出する.

表 4 に手法 1 にて生成される特徴ベクトルの例を示す.2 行目は対応する記号を示し,3 行目は出現頻度を表している. また,1 列目は入力項目数である.

3.2.2 手法 2

表 5 に手法 2 で抽出する特徴を示す. 手法 2 では手法 1 の特徴抽出法に加え, 英文字である a∼z,A∼Z と数字

```
/cdss9uiNbhir9putoDu/cBjtxYssdiNsg3Yetat/eYYEzPiVc/q_.iNv.php3?eogp0Lfl
4fl=bulk++insert++++od+++from+++++%27pwdump.exe%27++++++with++%28codepa
ge%3D%27RAW%27++%29&esonRewtqnidqi=hEPFQ
```

図 1 HTTP リクエストの例

表 5 特徴抽出方法 (手法 2)

特徴
入力項目数
各特殊記号の出現頻度
%エンコーディング
以外の特殊記号 (+ - .)
英数字の出現頻度

表 6 構成されているクラス

Class
Normal query(Valid)
XPATH Injection (XPathInjection)
OScommand Injection (OsCommanding)
LDAP Injection (LdapInjection)
SSI attacks (SSI)
SQL Injection (SqlInjection)
Directory-Traversal (PathTransversal)
Cross-Site Scripting (XSS)

である 0∼9 の計 62 個を特徴として加える ($x_{38} \sim x_{99}$). これらの英数字を特殊記号と同様に出現頻度を特徴量として抽出する.

4. 検知実験

本節では 3 章で説明した手法によってどれほどの精度になるのかを確認する検知実験について述べる. まず, 検知実験で使用するデータセットについて説明し, 次に, 深層学習に使用するネットワークの構造について述べる. 次に, 検知実験の流れを説明し, ネットワークの精度評価について述べる.

4.1 使用するデータセット

本実験では ECML/PKDD 2007 Discovery Challenge Dataset[6] を使用する. データセットは XML で定義されており, 各 HTTP リクエストはラベル付けで識別される.

このデータセットは特定の Web アプリケーションだけでなく, 様々な Web アプリケーションの HTTP リクエストで構成されている. データセットは 8 つのクラスがある. データセットを構成するクラスの種類は表 6 に示す. データセットには”GET”,”POST”,”PUT”の 3 つのメソッドがあるが, そのほとんどは”GET”メソッドである. 本実験では”GET”メソッドのみを用いる. また, 正常

表 7 手法 1 におけるネットワークの構造

層の形状	ユニット数	活性化関数
全結合	296	LeakyReLU
全結合	222	LeakyReLU
全結合	148	LeakyReLU
全結合	111	LeakyReLU
全結合	37	LeakyReLU
全結合	18	LeakyReLU
全結合	6	LeakyReLU
全結合	2	LeakyReLU
全結合	1	Sigmoid

表 8 手法 2 におけるネットワークの構造

層の形状	ユニット数	活性化関数
全結合	792	LeakyReLU
全結合	594	LeakyReLU
全結合	396	LeakyReLU
全結合	297	LeakyReLU
全結合	99	LeakyReLU
全結合	18	LeakyReLU
全結合	6	LeakyReLU
全結合	2	LeakyReLU
全結合	1	Sigmoid

なリクエスト 7436, 攻撃である異常なリクエスト 3752 を 1 つのセットとして取り出し, 学習用のデータと検証用のデータとする.

4.2 ディープニューラルネットワーク

ニューラルネットワークは, 人間の脳神経を模して作られた情報処理モデルである.[7] 入力層, 中間層, 出力層の 3 層からなるネットワークは多層マルチパーセプトロンと呼ばれる. このネットワークで中間層が複数になるものをディープニューラルネットワークと呼び, それを用いて行う機械学習を深層学習と呼ぶ.

手法 1 で用いたネットワークの構造を表 7 に, 手法 2 で用いたネットワークの構造を表 8 に示す.

層を重ねることで精度はわずかながら向上したが,10 層以上に重ねると精度の向上は見られなかったため, 入力層 1, 中間層 7, 出力層 1 からなる 9 層の構造とした. また、活性化関数に関しては ReLU よりも精度が高かったため, 入力層と中間層には LeakyReLU 関数を使用している. 出力は 0 ～ 1 の間にする必要があるので出力層には Sigmoid 関数を用いている.

表 9 混合行列

	予測は正常	予測は異常
実際は正常	TN	FP
実際は異常	FN	TP

4.3 実験方法

具体的な実験方法を図 2 に示す. 第 3.2 節で述べた各手法によって, 学習用データセットと検証用データセットから特徴ベクトルを抽出する. 抽出した特徴ベクトルの内, 学習用のものをネットワークに入力し, 学習させる. その後, 学習を終えたネットワークに検証用のデータを入力し, その出力から得られた分類結果と検証用データのラベルを比較することで, 精度を確認する.

本実験の評価方法は次節で述べる.

4.4 評価指標

ネットワークの精度は混合行列によって表として要約することが可能である. 表 9 に混合行列を示す. 各列はネットワークが予想したクラスを表し, 各行は実際のクラスを表している. この混合行列からは様々な評価指標を導くことができる.

本実験で使用する評価指標を表 10 に示す. 左の列には精度指標の名称を示しており, 右の列にはそれらの指標の計算式を示している.

5. 実験結果

本章では 4.1 節で述べたデータセットから,3.2 節で述べた手法 1 と手法 2 に従って作成した特徴ベクトルを用いて行った検知実験の結果を示す.

本実験では, 最適化関数を Adam, 損失関数を Binary

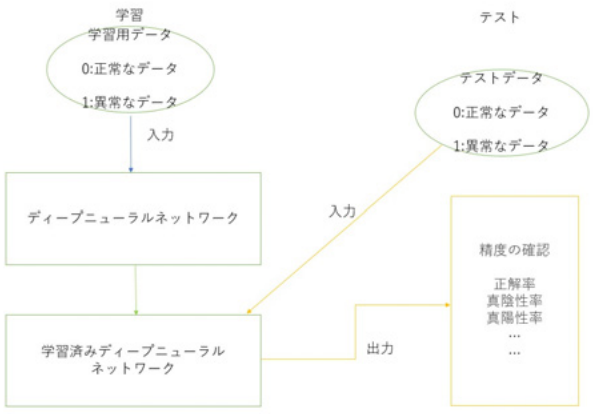

図 2 実験方法

表 10 評価指標

尺度	定義
真陰性数	TN
偽陽性数	FP
真陽性数	TP
偽陰性数	FN
正解率	$acc = \frac{TP+TN}{TN+FP+TP+FN}$
誤り率	$err = 1 - acc$
再現率	$recall = \frac{TP}{TP+FN}$
特異度	$specificity = \frac{TN}{FP+TN}$
陰性適中率	$NPV = \frac{TN}{TN+FN}$
適合率	$precision = \frac{TP}{TP+FP}$
F 値	$F - measure = \frac{2 \cdot recall \cdot precision}{recall+precision}$

表 11 手法 1 による実験結果

手法 1	
真陰性数	7406
偽陽性数	30
真陽性数	3114
偽陰性数	638
正解率	0.94
誤り率	0.05
再現率	0.82
特異度	0.99
陰性適中率	0.92
適合率	0.99
F 値	0.90

Cross Entropy にロジスティクス関数をかけたものを使用して学習を行った. また, ミニバッチ数を 50 とし,500 エポック学習させた.500 回以降でも損失値は減少したが, 精度の向上は見られなかったため, 本実験での学習回数は 500 エポックとした.

図 3 と図 4 にそれぞれの手法で抽出した特徴ベクトルを用いて深層学習を行った際の損失値を示す. どちらも十分に学習が収束していることが分かる.

表 11 に手法 1 による実験結果を, 表 12 に手法 2 に示す.

手法 1 による正解率は 0.94 であり, 手法 2 による正解率は 0.93 とわずかながら手法 1 の方が精度は高い結果となった.

偽陽性数は手法 1 の方が手法 2 に比べて少なく, 特異度, 陰性的中率, 適合率は手法 1 の方が優れている結果となった. 偽陰性数は手法 2 の方が少なく, 再現率は手法 2 の方が優れている結果となった. 全体的な精度を見ると, 手法 1 の方が精度が高い結果となった.

6. 考察

本章では特殊記号に着目した特徴ベクトル抽出方法である, 手法 1 と手法 2 を用いた実験結果の考察を示す.

手法 1 と手法 2 のそれぞれで抽出した特徴ベクトルをディープニューラルネットワークで学習させた結果, 損失値は減少している. つまり, 特殊記号に着目して抽出した特徴ベクトルで正常なリクエストと異常なリクエストの違いを学習できていると考えられる.

手法 1 での全体的な正解率は 94%, 手法 2 での全体的な正解率は 93%となっており, 特殊記号に着目して特徴を抽出する手法は有効であると考える.

また, 手法 1 の結果と手法 2 の結果を比較すると, 手法 1 の方が正常なリクエストを異常と誤検知する数が少なく, 評価指標の特異度, 陰性的中率, 適合率が高い. ただ, 手法 1 に比べて手法 2 の方がわずかながら異常なリクエストを正常なリクエストと分類してしまう数が少ない. しかし, それ以上に手法 2 では誤検知してし

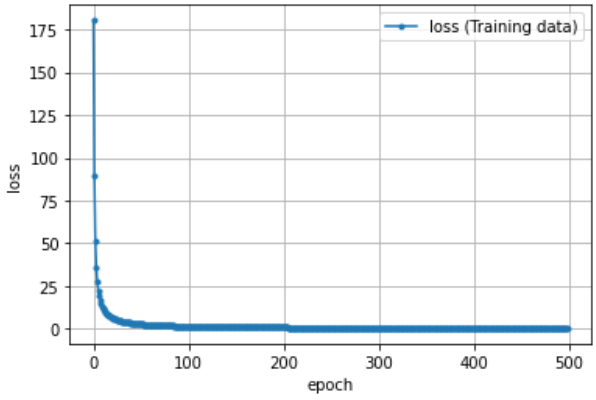

図 3 手法 1 における損失値の変化

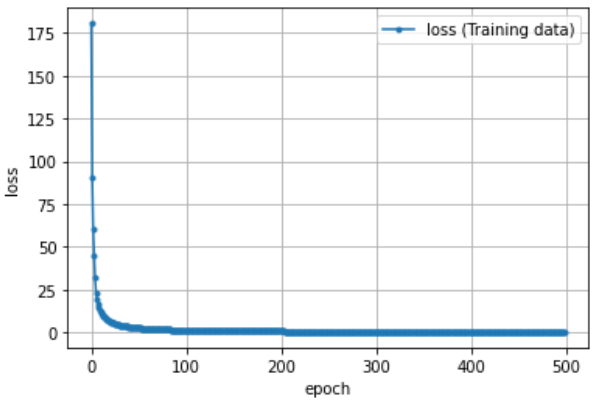

図 4 手法 2 における損失値の変化

表 12 手法 2 による実験結果

手法 2	
真陰性数	7275
偽陽性数	161
真陽性数	3133
偽陰性数	619
正解率	0.93
誤り率	0.06
再現率	0.83
特異度	0.97
陰性適中率	0.91
適合率	0.95
F 値	0.88

まっている.F 値を見ると, 手法 1 の方が高く, 全体的には手法 1 の方が有効であることが分かる. つまり, 特殊記号の出現頻度は特徴として有効であるが, 英数字の出現頻度は特徴としてそれほど有効ではない考える.

手法 1, 手法 2 のどちらの手法においても偽陽性は少なくなっており, 特異度や適合率は 95%を越えている. そのことから, 本研究で提案した手法は正常なリクエストを異常なリクエストであると誤検知することは十分に少ないことが分かる. しかし, 再現率を見ると, どちらも 80%ほどと検知器としてしようするにはまだまだ低い結果となっている. 再現率は全ての異常なリクエストの内, 異常なリクエストであると分類できた割合である. つまり, 本研究で提案した手法では, 正常なリクエストを正常であると分類することは十分にできるが, 異常なリクエストを正常であると判断してしまう可能性がそれなりにあると言える. 陰性的中率が 90%ほどなので, 正常であると分類されたリクエストの内,1 割ほどは異常なリクエストとなってしまっている. 再現率が低い状態のままでは検出器として運用するのは難しいため, 偽陰性数を抑え, 再現率を向上させるための改善が必要である.

異常リクエストを正常リクエストと分類してしまった攻撃は OS コマンドインジェクションが特に多く, 次いでディレクトリ・トラバーサルであった.

OS コマンドインジェクションは 2.1 節でも述べたが, 不正な OS コマンドを注入することで, 対象の PC やサーバにアクセスする攻撃である. また, ディレクトリ・トラバーサルはディレクトリの相対パスを注入することで, 本来は閲覧できないはずのデータを不正に入手する攻撃である. この 2 つの攻撃に共通する特殊記号は「/」と「.」である. これらの特殊記号を多く含む攻撃は正常なリクエストとの見分けがあまりついていないために, この 2 つの攻撃を正常なリクエストと分類してしまっていると考える. また, この 2 つの攻撃には他の攻撃と重なる特殊記号が少ないのも, あまり分類できない一因ではないかと考える.

ネットワークの学習は十分に行われており, 正常なリクエストと異常なリクエストを分類する閾値を 0 という厳しい数字を使用しているにも関わらず,OS コマンドインジェクションをおよそ半分ほどしか異常と分類できていない. また, 英数字の出現頻度を特徴ベクトルに加えた手法 2 であっても,SQL インジェクションやディレクトリ・トラバーサルを正常なリクエストであると分類してしまう数は減ったものの,OS コマンドインジェクションにはあまり影響は無かった. 本研究の提案手法では OS コマンドインジェクションを満足に分類できないと考える.

再現率を向上させるためには, ネットワークの構造を変更する, 特殊記号に加えて新たな特徴を抽出するなど, 特に OS コマンドインジェクションを見分けられる方法を考える必要がある.

7. 結　言

本章では Web への攻撃を検知するために提案した特徴抽出手法について総括し, 今後の課題について述べる.

本研究では様々な Web 攻撃手法に共通する特徴を用いて, それらの攻撃を網羅的に検知する事を目的とした.

本研究では攻撃用の異常リクエストには特殊記号が多く出現することに着目したアプローチに沿って,2 つの特徴抽出手法を提案した.

提案した特徴抽出手法の有用性を確認するために ECML/PKDD 2007 Discovery Challenge Dataset を用いて実験を行った. 実験の結果, ある程度の分類は可能であると実証された.

次に, 今後の課題について述べる.

まず, 本研究で深層学習に用いたディープニューラルネットワークの改善が必要である。現状では, ある程度の分類は可能である. しかし, 検知器として使用するには再現率が低い. 本研究で用いたディープニューラルネットワークは全て全結合層からなるものであったが, ネットワークには畳み込みなどの種類がある. それらを使用することで精度改善が期待できる.

また、特徴抽出手法の改善も必要であると考える.

上記のように深層学習に使うネットワークを変えることで, より特徴を掴める可能性は存在するが, 現状, OS コマンドインジェクションのリクエストと正常リクエストを見分けられていない. つまり, 本研究で提案した特徴抽出手法により作成した特徴ベクトルでは完全に

分類できていない. また, 英数字の出現頻度を特徴ベクトルに加えても, 精度は向上しないことが分かっている.

本研究による提案手法では全ての特殊記号を特徴量としたが, 正常リクエストと異常リクエストの分類にあまり役立っていない特殊記号を省いて抽出する. あるいは, 出現頻度だけではなく, 出現する順番も特徴ベクトルで表現できるようにするといった改善点が考えられる.

上記の課題を取り入れることで, より高い精度での分類が可能になると期待できる.

参考文献

[1] 情報処理推進機構:ソフトウェア等の脆弱性関連情報に関する届出状況 [2021 年第 2 四半期（4 月～6 月）],<https://www.ipa.go.jp/files/000092739.pdf>(2021/10)

[2] JVN iPedia 脆弱性対策情報データベース:Apache HTTP Server におけるディレクトリトラバーサルの脆弱性,<https://jvndb.jvn.jp/ja/contents/2021/JVNDB-2021-000090.html>(2021/10)

[3] 情報処理推進機構:安全なウェブサイトの作り方改訂第 7 版,<https://www.ipa.go.jp/files/000017316.pdf>(2021/10)

[4] OWASP Top 10 2021,<https://owasp.org/Top10/ja/>(2021/10)

[5] 清水大貴, 小高知宏, 黒岩丈介, 諏訪いずみ,and 白井治彦:機械学習を用いた Web アプリケーション攻撃検知手法の提案, 福井大工報, 68, 51-58 (2020).

[6] Analyzing web traffic:Ecml/pkdd 2007 discovery challenge, <http://www.lirmm.fr/pkdd2007-challenge/>(2021/4)

[7] 麻生英樹: 多層ニューラルネットワークによる深層表現の学習. 人工知能学会誌, 28-4,649–659(2013).