情報処理学会論文誌 Vol.56 No.9 1826–1833 (Sep. 2015)

攻撃特徴記号に基づくWAF開発

園田 道夫[1] 松田 健[2,a)]

受付日 2014年11月28日, 採録日 2015年6月5日

概要：Webアプリケーションをターゲットとするサイバー攻撃は，Webアプリケーションを運営管理する者だけでなく，その利用者にとっても大きな脅威となっている．このような脅威からWebアプリケーションを護る方法としてWebアプリケーションファイアウォール（WAF）が開発され，実際に運用されている．本研究では，潜在曲線モデルを応用することで攻撃データと正常データに含まれる記号に重みを付与し，SQLインジェクション攻撃の特徴を抽出して攻撃検出する手法を提案した．提案手法とApacheのモジュールであるModSecurityとの検出率を比較した結果，攻撃検出についてはModSecurityの方が提案手法よりやや攻撃検出率が高く，正常検出については提案手法の方がModSecurityより正常検出率がかなり高くなることを確認した．さらに，提案手法を実装したSQLインジェクション攻撃を検出するApacheモジュールの一部をWebページ上に公開した．

キーワード：WAF，攻撃検出，潜在曲線モデル，ModSecurity

Web Application Firewall Development Based on Attack Feature Symbols

Michio Sonoda[1] Takeshi Matsuda[2,a)]

Received: November 28, 2014, Accepted: June 5, 2015

Abstract: Cyber attacks which target Web applications are becoming big threat against the manager and coustmer of Web application driven a database. As the method of prevention and detection for Web applications, Web application firewall (WAF) had been developed and used in real. In this study, we proposed a new SQL injection attacks detection method by giving a weight to symbols of input strings. Moreover, we developed a module of Apache for detecting SQL injection attacks. To check the effectiveness of our proposed method, we compared the detection performance our proposed method and ModSecurity which is the module of Apache. As the result, the detection performance of ModSecurity was little good than our proposed method for attack data. However, the detection performance of our proposed method was better than ModSecurity for normal data. Our developed module had opened on our Web page.

Keywords: WAF, attack detection, latent curve model, ModSecurity

1. はじめに

急速なICT技術の高度化・多様化にともない，データベース駆動型のWebサービスを利用するユーザは増加の一途をたどっている．商用のWebサービスでは，データベースに個人情報が格納されているため，Webサービスを実現するためのWebアプリケーションが持つ脆弱性に対する攻撃（Webアプリケーション攻撃）には十分な対策が必要となる．

Webアプリケーション攻撃については，OWASP（Open Web Application Security Project）によってWebアプリケーションに関する脆弱性のうち，攻撃者に狙われやすい脆弱性がランク付け（OWASP Top 10）されており，本研究で取り扱うSQLインジェクションはその中でも重要な脆弱性として紹介されている[1]．

SQLインジェクション攻撃は悪意のあるSQL文法を埋

[1] 中央大学大学院理工学研究科
Bunkyo, Tokyo 112–8551, Japan
[2] 静岡理工科大学総合情報学部
Fukuroi, Shizuoka 437–8555, Japan
[a)] tmatsuda@cs.sist.ac.jp

め込む手法の攻撃であり，攻撃が成功するとWebアプリケーションのデータベースを不正に操作することが可能となる．SQLインジェクション攻撃からWebアプリケーションを守るための技術はこれまでにも多くの手法が開発・運用されている [2]．たとえば，自動静的分析ツールによってWebアプリケーションを構成するプログラムのソースコードを解析することで脆弱性を検出する方法 [3]，入力データからSQL文を組み立てる際に，文字列連結演算でなくプリペアドステートメントとバインド機構を使用することで根本的にSQLインジェクションの脆弱性を解消する方法などが開発されている [4]．プリペアドステートメントによって，同じパターンのSQL文を繰り返し使用する場合はアプリケーションの処理に関するパフォーマンスの向上が期待される反面，同じパターンでないSQL文を複数同時に使用する場合などにはパフォーマンスの低下を招く場合もある．上述のようなSQLインジェクション攻撃を原理的に引き起こさせないような対策のほか，webアプリケーションに入力される文字列を確認して攻撃を検出するWebアプリケーションファイアウォール（WAF）が開発され，実際に運用されている [5]．WAFの攻撃検出方法は，ブラックリストやホワイトリストによるリスティング形式，正常な入力パターンを学習するパターン認識であり，最近ではSVMやベイジアンネットワークなどの機械学習の手法を取り入れて開発されたWAFも運用されるようになっている [6], [7], [8], [9]．機械学習の手法を用いてWAFを開発するメリットとして，未知の攻撃や予期せぬ脆弱性に対応できる可能性があげられ，これにより攻撃検出のために参照するデータの量を減らし，プログラムの処理効率を高めることができると考えられる．

SQLインジェクション攻撃については，正常な入力データと比較すると攻撃データに多くの特殊記号が含まれる傾向があるため，WAFを用いた攻撃検出は有効であると考えられる．しかしながら，記号を多く含む攻撃でない入力データ（正常データ）も存在するため，既存の攻撃検出法では，このような正常データを攻撃データと誤検出する可能性を排除することは容易でないと考えられる．

本研究では，潜在曲線モデルの手法 [10] を応用することで記号に重みを付与し，その情報を用いてSQLインジェクション攻撃を検出する手法を提案した．記号の重みは攻撃データと正常データを収集してそれぞれのデータの集合を考え，入力文字列に含まれる記号と文字列におけるその記号の含有率に対応する座標をユークリッド空間上にとることで攻撃と正常の両方の特徴を表現する数学モデル（潜在曲線モデルの考えを応用したもの）を構成する．攻撃検出には，攻撃データから座標の組を作成し，数学モデルとの残差を計算する．提案手法の有用性を検証するために，特徴抽出に使用したデータを含まないテスト用の攻撃データと正常データを用意し，それらのデータを提案手法を用いて開発した攻撃検出モジュールとModSecurity [11] を用いて検出する実験を行った．実験により，攻撃データの検出率についてはModSecurityが提案手法を上回り，正常データの検出率については提案手法がModSecurityを上回るという結果が得られた．

以下，2章ではSQLインジェクション攻撃の概要について紹介する．3章ではSQLインジェクション攻撃の検出に関する従来研究をまとめた．4章ではSQLインジェクション攻撃の特徴を抽出するアルゴリズムを，5章では攻撃を検出するアルゴリズムを提案し，検出実験を行った．6章では検出実験に関する考察を行い，最後の7章でまとめを行う．

2. SQLインジェクション攻撃

本章では，SQLインジェクション攻撃の概要について紹介する．SQLインジェクション攻撃は，ブラウザの入力欄からWebアプリケーションに対して想定外の入力データが送り込まれることで発生するWebアプリケーション攻撃である．たとえば，あるWebアプリケーションを利用する際に，URLにユーザを特定する情報が含まれている場合を考える．ここでは，ユーザの入力文字列からデータベースへアクセスするためのSQL文を直接作成するWebアプリケーションを想定する．

http://www.sqlinjection.ac.jp?id=14990000

これによりWebサーバはアプリケーションサーバにid=14990000という情報を送り，データベースに送るためのSQL文を作成する．その結果，id=14990000に紐づいたユーザの情報をWebページに表示することができる．ここでURLを

http://www.sqlinjection.ac.jp?id=14990000’ or ’A’=’A

と変更すると，A=Aはつねに真であるため，id=14990000に紐づいた情報も含めて，データベース上のすべてのユーザの情報がWebページに表示される危険性を持つことになる．このようにSQLインジェクション攻撃に対して脆弱なWebアプリケーションでは，任意のSQL文をブラウザの入力欄から入力されることによってデータベースへの不正なアクセスを許すことになってしまう．

SQLインジェクション攻撃を実現するには，シングルクォートやセミコロンなどの特殊な記号が必要となるため，Webアプリケーションに対する通常の入力（正常データ）とは異なる特徴を持っている．しかしながら，ブラウザ上の入力欄に入力される文字列は，たとえば，英文，顔文字，Wikiの文法である場合も考えられるため，SQLインジェクション攻撃に含まれる記号を利用した攻撃検出法では，正常データを攻撃データと誤検出することをどのように防ぐかということが問題となる．

著者らは，SQLインジェクション攻撃に含まれる記号に着目した攻撃検出法を提案し [12]，攻撃データを複数個収

集して SQL インジェクション攻撃に含まれる記号の分布を調べると，記号の分布がベキ乗則に従う傾向にある [13] ことを確認している．

本研究では，文献 [13] の手法に基づいて攻撃データの特徴と正常データの特徴をそれぞれのサンプルを収集・比較することで抽出し，攻撃検出を行うアプリケーションを Apache のモジュール開発機能を用いて開発した．

3. 従来研究

SQL インジェクション攻撃の基本的な対策は入力の制限，またはバインド機構を利用することである．しかしながら，これらの対策が十分でない場合，入力文字列からそれが攻撃か否か判断する作業が必要である．このような攻撃検出では，過去の攻撃パターンを応用することが一般的である．最も単純な方法は過去に行われた攻撃のデータからブラックリストを作成したり，パターンマッチングを行ったりするものであり，この方法は Apache のモジュールである ModSecurity にも正規表現などを用いて応用されている．

しかしながら，これらの手法では未知の攻撃に対応することは難しい．そこで，未知の攻撃に対応するための方法として機械学習の手法を応用した攻撃検出法が提案されている [6], [7], [8]．著者らも，SVM やナイーブベイズなどの機械学習を用いた攻撃検出法について検討しており [14], [15]，その有用性を確認している．しかしながら，機械学習を用いて攻撃検出する際，攻撃を正常と判別したり，正常を攻撃と判断したりする誤検出の問題がしばしば起こる．SQL インジェクション攻撃には特殊記号が含まれることは周知の事実であるが，それらの特殊記号を用いて攻撃検出を行う際，正常データにもそれらの記号が含まれている場合に誤検出が起きやすい．たとえば，Wiki の文法や顔文字は明らかに SQL インジェクション攻撃ではないが，SQL インジェクション攻撃に多用される特殊記号が含まれる場合があり，それらの記号が原因で誤検出を導くことがある．

ほとんどの SQL インジェクション攻撃は特殊記号を含むため，それらに注意することで攻撃を検出することは比較的容易であるが，正常を攻撃と誤検出することを考えたトレードオフをどのように設定するかということが誤検出を少なくすることで重要であるといえる．攻撃検出は，攻撃か正常かというラベルが分からない（観測されない）データにラベルをつける作業であり，そのラベルは観測されるデータである入力文字列によって与えられるものである．したがって，攻撃検出には，攻撃や正常というラベルデータを潜在変数とするモデルが自然にあてはまる．

本研究では，潜在曲線モデルと呼ばれる構造方程式モデルを用いて SQL インジェクション攻撃を検出する方法を提案し，それに基づいた攻撃検出を行う WAF を開発した．

4. 特徴抽出アルゴリズム

著者らは，SQL インジェクション攻撃に含まれる記号の特徴をベキ乗則の一種であるゼータ分布に基づいて抽出している．本研究では，文献 [18] の手法を応用し，潜在曲線モデルを用いて攻撃データと正常データのサンプル集合を比較することでそれぞれの特徴を抽出する．

4.1 特徴抽出アルゴリズム

ここでは，攻撃データと正常データの特徴を抽出するためのアルゴリズムを提案する．

［Step 1］
攻撃データを I_A 個，正常データを I_N 個準備する．

［Step 2］
攻撃，正常それぞれのデータの集合に対し，出現するすべての記号（攻撃データ集合については J_A 個，正常データ集合については J_N 個とする）とその頻度を計算し，記号を出現頻度の大きい順に並べ替え，その記号を順に $s_1, s_2, \cdots, s_J$ とする．

データ l_i に含まれる記号 s_j の個数を $z_i(s_j)$ とおくと，攻撃データ集合における記号の頻度分布は，$j = 1, 2, \cdots, J_A$ に対して

$$T_A(s_j) = \sum_{i=1}^{I_A} \frac{z_i(s_j)}{|l_i| \cdot I_A}$$

正常データ集合における記号の頻度分布は，$j = 1, 2, \cdots, J_N$ に対して

$$T_N(s_j) = \sum_{i=1}^{I_N} \frac{z_i(s_j)}{|l_i| \cdot I_N}$$

と表すことができる．ここで，l_i は文字列のデータ，$|l_i|$ は文字列 l_i に含まれる記号の総数を表すものとする．本研究では，攻撃データ集合における記号の出現頻度分布のうち，出現頻度が大きい記号から順に x 個の記号を利用して攻撃の特徴を抽出する．

［Step 3］
攻撃データと正常データを 3 次元ユークリッド空間 $\mathbf{R^3}$ 上の点 (x, y, t) として表現する．$\mathbf{R^3}$ の第 1 座標 X は，以下のように攻撃データ集合に出現する記号に対応させるように定義する．具体的には，攻撃データ集合の記号出現頻度分布から上位 x 個のデータに着目し，それらを $s_1, s_2, \cdots, s_x$ とする．この x 個の記号の列を第 1 座標 X として

$$1 = s_1, 2 = s_2, \cdots, x = s_x$$

と表すことにする．これにより，記号 s_j（$j = 1, 2, \cdots, x$）

を，$\mathbf{R^3}$ の X 軸の $(j,0,0)$ という座標に対応させることができる．次に，第 2 座標の値を，第 1 座標 j に対応する記号 s_j から $y_j(l_i) = \frac{z_i(s_j)}{|l_i|}$ を計算した値として定義する．ここまでの定義により，もし文字列 l に記号 s_{j_1}，s_{j_2} が含まれる場合，座標の組 $(j_1, y_{j_1}(l))$ と $(j_2, y_{j_2}(l))$ が得られる．最後に，第 3 座標 t の値は，データが攻撃である場合は $t = 1$，正常である場合は $t = 0$ とする．

［Step 4］
前の［Step 3］で作成したデータを用いて，以下の潜在曲線モデルのパラメータを最尤法で推定する．

$$y = a_0 + a_1 x + a_2 x^2 + \epsilon_1$$
$$a_0 = b_{00} + b_{10} t + \epsilon_2$$
$$a_1 = b_{01} + b_{11} t + \epsilon_3$$
$$a_2 = b_{02} + b_{12} t + \epsilon_4$$

与えられたデータから，パラメータ b_{00}，b_{10}，b_{01}，b_{11}，b_{02}，b_{12} を推定することが目標である．ただし，ϵ_i $(i = 1, 2, 3, 4)$ は平均 0，分散 $\sigma_i^2 > 0$ の正規分布に従う誤差項である．**図 1** から SQL インジェクション攻撃には出現頻度が極端に高い記号がいくつか存在することが分かり，これらの記号は攻撃検出に有用であると考えられる．出現頻度の高い部分を表現するモデルとして最も単純なものは上記の $y = a_0 + a_1 x + a_2 x^2$ である．しかしながらこのモデルは，$a_2 > 0$ の場合，ある x を境目に微分係数 y' がある一定の正の数より大きくなるため，攻撃検出には微分係数が 0.152 より小さくなるする区間に含まれる記号のみを利用する．なお，0.152 という値は，**図 3** において $x = 5$ のときの微分係数 y' の値がおよそ 0.151 となるために設定した数値である．このようにすることでモデル $y = a_0 + a_1 x + a_2 x^2$ は x の値が大きくなると y の値は小さくなり，x の値が小さい（攻撃に頻出する記号に対応する）ときに y の値が大きくなるため，y の値を攻撃特徴の重みとして使用することができる．

4.2 SQL インジェクションの攻撃特徴

ここでは，提案手法である特徴抽出アルゴリズムを用いて SQL インジェクション攻撃の特徴抽出を行う．

［Step 1］本研究で使用するサンプルは以下のとおりである．攻撃データのサンプルは文献 [16] から収集し，そこから攻撃データ再構成する攻撃データ生成機を作成して 2,779 個（$J_A = 2779$）の攻撃データを準備した．正常データのサンプルはブラウザに入力される個人情報に関連する仮想的なデータを用意し，顔文字や Wiki の文法など，特殊な記号を多く含む文字列を 444 個（$J_N = 444$）の正常データを準備した．

表 1 横軸（x 軸）に対応する 5 つの記号
Table 1 Five symbols on the horizontal axis.

記号	Space	’	;	)	(
x 座標	1	2	3	4	5

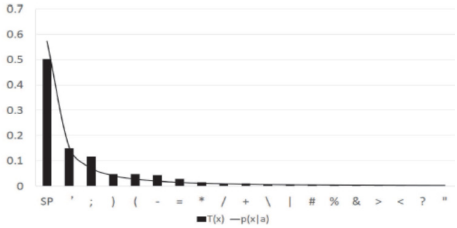

図 1 攻撃データの記号分布
Fig. 1 Distribution of symbols on attack data set.

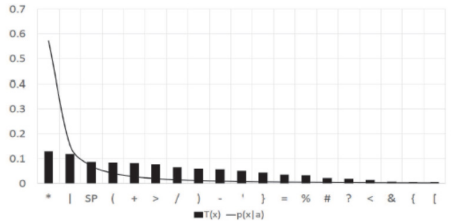

図 2 正常データの記号分布
Fig. 2 Distribution of symbols on normal data set.

［Step 2］2,779 個の攻撃データ集合に含まれている記号のうち，出現頻度の高い 5 つの記号（$x = 5$ とした）を選ぶと**表 1** のようになる．

本研究では，表 1 にある 5 つの文字を用いて SQL インジェクション攻撃の特徴抽出を行う．参考までに，攻撃と正常それぞれのデータ集合に含まれる記号の分布（出現頻度の総和で正規化したもの）は図 1，**図 2** のようになる．ここで，図 1，図 2 の曲線は，文献 [13] の手法に基づいて計算したゼータ分布を表しており，図の棒グラフのデータからゼータ分布のパラメータを推定して計算したものである．

［Step 3］，［Step 4］本研究では 2,799 個の攻撃データと 444 個の正常データのサンプルからそれぞれのデータ集合を作成し特徴抽出を行った．図 3，**図 4** はこれらのデータ集合から計算したモデル $y = (b_{00} + b_{10} t) + (b_{01} + b_{11t}) x + (b_{02} + b_{12} t) x^2$ の推定結果である．図 3, 図 4 の点は［Step 3］で構成したデータを表し，曲線は［Step 4］での推定結果のモデル（2 次関数）を表している．なお，これらの図の横軸（x 軸）は表 1 に示した記号に対応する．

表 2 は潜在曲線モデルから得られる記号の重みの算出結果である．

5. 実験

前章では，潜在曲線モデルを応用することで SQL イン

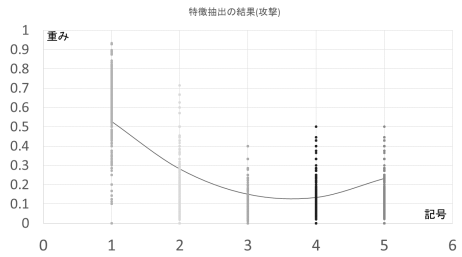

図 3 学習用データ（攻撃 $t=1$）の分布と推定結果
Fig. 3 Distribution of the learning data (attack $t=1$) and the estimation result.

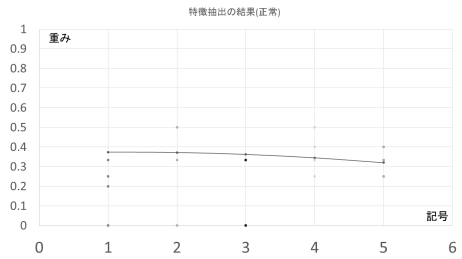

図 4 学習用データ（正常 $t=0$）の分布と推定結果
Fig. 4 Distribution of the learning data (normal $t=0$) and the estimation result.

表 2 出現記号と重み
Table 2 Appearance symbols and their weight.

記号	Space	’	;	)	(
重み（攻撃）	0.5274	0.2815	0.1503	0.1339	0.2323
重み（正常）	0.3732	0.3712	0.3616	0.3444	0.3196

ジェクション攻撃の特徴抽出を行うアルゴリズム提案した．本章では，攻撃検出のためのアルゴリズムを提案し，実験を行う．

5.1 攻撃検出アルゴリズム

［Step A］
入力文字列に含まれる文字 s_{j_k}（$k=1,2,\cdots,K$）を求め，座標の組

$$T_N=\{(j_k,y_{j_k},0)\}_{k=1}^{K}, T_A=\{(j_k,y_{j_k},1)\}_{k=1}^{K}$$

を作る．なお，記号 s_j が入力文字列に含まれない場合，そのときの y_j の値は欠損値として扱う．ただし本提案手法では，5 つの特徴記号のうち 3 つ以上の記号が出現しないデータについてはパラメータが一意に定まらないため，この条件を満足するデータを学習データとして使用する．なお，検証用のテストデータには上記のようなデータの制限を与えないため，仮想的に学習段階では出現しなかった未知のデータが含まれていることになっている．

［Step B］
前の Step で作成した座標の組のうち，どちらの方が特徴抽出アルゴリズムで求めた潜在曲線モデルの超平面にあてはまりが良いか調べる．あてはまりの良さを調べる指標として残差を利用する．具体的には，座標の組 $T_A=\{(j_k,y_{j_k},1)\}_{k=1}^{K}$ の残差 $e(T_A)$ が $T_N=\{(j_k,y_{j_k},0)\}_{k=1}^{K}$ の残差 $e(T_N)$ より小さいときは入力された文字列を攻撃と判定し，それ以外のときは正常と判断する．

5.2 データ

5.2.1 攻撃データ

提案手法の有用性を検証するための攻撃データを文献 [16] から 200 個収集した．なお，これらのデータは特徴抽出の際には使用していないことに注意する．以下は検証のために使用したデータの一部である．

l_1 : id=3 union all select 1,2,3,4,5 from admin/*
l_2 : SELECT ’A’ || ’B’ FROM dual;
1) or (’ab’ = ’a”b
SELECT UTL_INADDR.get_host_address FROM dual;
l_3 : %’SELECT CHR(65)||CHR(66);

これらのデータから得られる座標の組はそれぞれ以下のように表現される．ただし，$y=0$ であるときは欠損データとして扱う．

l_1 :
$T_{A_1}=\{(1,1,1)\}$
$T_{N_1}=\{(1,1,0)\}$
l_2 :
$T_{A_2}=\{(1,0.48,1),(2,0.36,1),(3,0.08,1),(4,0.04,1),(5,0.04,1)\}$
$T_{N_2}=\{(1,0.48,0),(2,0.36,0),(3,0.08,0),(4,0.04,0),(5,0.04,0)\}$
l_3 :
$T_{A_3}=\{(1,0.143,1),(2,0.143,1),(3,0.143,1),(4,0.286,1),(5,0.286,1)\}$
$T_{N_3}=\{(1,0.143,0),(2,0.143,0),(3,0.143,0),(4,0.286,0),(5,0.286,0)\}$

これらの座標の組と，特徴抽出アルゴリズムによって得られた超平面との残差はそれぞれ以下のとおりである．

l_1 :
$e(T_{A_1})=0.2233$
$e(T_{N_1})=0.3929$
l_2 :
$e(T_{A_2})=0.0592$
$e(T_{N_2})=0.2616$
l_3 :
$e(T_{A_3})=0.1931$
$e(T_{N_3})=0.1576$

残差の計算結果から，$i=1,2,3$ のとき $e(T_{A_i})<e(T_{N_i})$ であるから，l_1，l_2，l_3 は攻撃として正しく検出されること

が分かる．

5.2.2 正常データ

SQL インジェクション攻撃には多くの記号が含まれる．したがって，記号を多く含む入力文字列は，SQL インジェクション攻撃と似たような特徴を持っている可能性がある．そこで，提案モデルの有用性を検証するために，本研究では記号が多用されている正常なパラメータを含むデータを 50 個収集・作成し，正常データとして使用した．以下に，サンプルの一部を示す．

l_4 :
https://www.google.co.jp/search?q=%22%3Balert(document.domain)%2F%2F&oq=%22%3Balert(document.domain)%2F%2F&aqs=chrome..69i57&sourceid=chrome&es_sm=122&ie=UTF-8
l_5 :
http://msdn.microsoft.com/ja-jp/library/kk6xf663(v=vs.80).aspx
これらのデータから得られる座標の組はそれぞれ以下のように表現される．

l_4 :
$T_{A_4} = \{(4, 0.5, 1), (5, 0.5, 1)\}$
$T_{N_4} = \{(4, 0.5, 0), (5, 0.5, 0)\}$
l_5 :
$T_{A_5} = \{(4, 0.5, 1), (5, 0.5, 1)\}$
$T_{N_5} = \{(4, 0.5, 0), (5, 0.5, 0)\}$

この例では，l_4，l_5 ともに同じ座標の組となっている．このように，異なるデータでも特徴を表現する座標は同じものになる場合が存在する．これらの座標の組と，特徴抽出アルゴリズムによって得られた超平面との残差はそれぞれ以下のとおりである．

l_4 :
$e(T_{A_4}) = 0.2057$
$e(T_{N_4}) = 0.0568$
l_5 :
$e(T_{A_5}) = 0.2057$
$e(T_{N_5}) = 0.0568$

残差の計算結果から，$i = 4, 5$ のとき $e(T_{A_i}) \geq e(T_{N_i})$ であるから，l_4，l_5 は正常として正しく検出されることが分かる．

5.3 検出結果

表 3 は，提案手法を用いて検出実験を行った結果まとめたものである．特徴抽出アルゴリズムから得られたモデルは

$$y = (b_{00} + b_{10}t) + (b_{01} + b_{11}t)x + (b_{02} + b_{12}t)x^2$$

表 3 実験結果（提案手法）
Table 3 Experimental result (proposed method).

	攻撃	正常
学習用データ	2,779 個	444 個
テスト用データ	200 個	50 個
検出率	96%	100%

表 4 実験結果（ModSecurity）
Table 4 Experimental result (ModSecurity).

	攻撃	正常
テスト用データ	200 個	50 個
検出率	100%	38%

表 5 実験結果（SVM）
Table 5 Experimental result (SVM).

	攻撃	正常
学習用データ	2,779 個	444 個
テスト用データ	200 個	50 個
検出率	100%	0%

$b_{00} = 0.368$
$b_{10} = 0.521$
$b_{01} = 0.009$
$b_{11} = -0.428$
$b_{02} = -0.004$
$b_{12} = 0.061$

である．このモデルは，$t = 1$ のときは攻撃の特徴を，$t = 0$ のときは正常の特徴を表している．

なお，提案手法の有用性を確認するため，Apache のモジュールである ModSecurity に対して上記実験と同様のテスト用データを適用し，検出実験を行った．ModSecurity は OWASP ModSecurity Core Rule Set [17] に基づいて攻撃を検出し，攻撃から Web アプリケーションを守るために広く利用されているだけでなく，攻撃検出のためのルール更新が頻繁に行われているという特徴を持つ．

その結果は**表 4** のとおりである．

文献 [15] では，機械学習のアルゴリズムである SVM を用いた攻撃検出について検討している．本研究では提案手法と機械学習を用いて攻撃検出する手法を比較するため，表 2 にある 5 つの記号を特徴記号として文献 [15] の攻撃検出手法を用いて実験を行った．結果は**表 5** のとおりである．

なお，計算には R の library（kernlab）を使用し，ガウスカーネルのパラメータを以下のように設定した．

ksvm(V6 ., data=x, type = ”C-bsvc”, kernel = ”rbfdot”, kpar= list(sigma = 10))

実験結果については次の章で考察を行うが，カーネル

トリックを用いたSVMでは正常をすべて攻撃と判断しているため，ここで簡潔に考察を述べる．この実験では，提案手法とその他の手法の比較を行うため，5.1節の［Step A］で述べたとおり，提案手法のパラメータ推定が計算的に可能なデータのみを用いて学習を行った．そのため，学習データには正常なデータであっても3種類以上の攻撃特徴記号を含むという性質を持っている．一方，テスト用のデータには攻撃特徴記号を含まない正常データもそうでないデータも含まれているため，表5のような正常データにおける誤検出を引き起こしたものと考えられる．

6. 考察

本章では，前章の実験結果について考察する．実験では，提案手法を用いて開発したApacheモジュールと，Apacheのモジュールである ModSecurity を用いて攻撃データと正常データの検出を行った．なお，ModSecurity はデフォルトの状態で使用し，以下の環境のもとで検出実験を行った．

CentOS6.6
Apache/2.2.15
PHP 5.5.18
FirefoxESR 31.2.0
mysql Ver 14.14 Distrib 5.5.40, for Linux (x86_64)
Core ModSecurity Rule Set ver.2.2.6
mod_security-2.7.3-3.el6.x86_64

以下，攻撃データと正常データそれぞれの実験結果について考察を行う．

6.1 攻撃検出について

ModSecurity は用意した200個すべての攻撃データを攻撃と正しく検出した．一方，提案手法では，188個の攻撃データを攻撃と検出し，残りの12個のデータを正常と誤検出した．提案手法が誤検出したデータを1つ紹介する．

|SELECT current_setting('data_directory');

表2にある記号に付与される重みの情報を見ると，この文字列に含まれる記号の重みは攻撃・正常ともに大きくなると考えられる．したがって，提案手法はこのデータを正常データとして検出したと考えられる．この例は攻撃データであるから攻撃データと検出すべきであるが，ModSecurity では記号を多く含む文字列は攻撃として検出する可能性が非常に高いと考えられる．たとえば，URLのパラメータに記号が多用されるデータを攻撃と検出する可能性が高い．一方，提案手法は ModSecurity や SVM を用いた検出法と比較すると，入力文字列における記号の扱い方が柔軟であると考えられる．そのため，攻撃を正常と誤検出する危険性がある一方，記号が多用されているデータをいつでも攻撃と検出しないことから正常データを攻撃と誤検出する危険性は緩和されるものと考えられる．

6.2 正常検出について

提案手法は用意した50個すべての正常データを正常と正しく検出した．SVM を用いた検出方法では，5.3節で述べたとおり，すべての正常データを正常と正しく検出することができなかった．一方，ModSecurity は，19個の正常データを正常と検出し，残りの31個のデータを攻撃と誤検出した．ModSecurity が誤検出したデータを1つ紹介する．

https://www.google.co.jp/?gws_rd=ssl#q=
(%EF%BE%9F%E2%88%87%5E*)%EF%BD%B5
%EF%BE%8A%EF%BE%96%E2%99%AA

この例が示すとおり，SQL インジェクション攻撃にはなりえない，記号を多く含むデータに対しても ModSecurity（デフォルト設定）は攻撃として検出することが分かる．それに対して，提案手法では記号それぞれに対して攻撃の重みと正常の重みを付与し，それらの情報を抽象化（数学モデル化）して攻撃を検出するため，攻撃検出のパフォーマンスは ModSecurity より劣る可能性は高い一方，正常データを攻撃と誤検出する可能性は低くできるものと考えられる．

しかしながら提案モデルを用いて検出を行う場合はいくつか注意すべき点がある．記号の重みは攻撃データ集合と正常データ集合それぞれの集合における記号の出現頻度から計算されるため，サンプルとなるデータの集め方に依存してしまう．したがって，特徴抽出にはデータの収集方法が重要となるが，一方で，記号を多く含む正常データを許容する場合，そのようなデータを収集・作成することで記号を多く含む入力文字列に対して柔軟な攻撃検出を実現できるものと考えられる．

6.3 モデルについて

本研究では，潜在曲線モデルの考え方を応用することでSQL インジェクション攻撃の特徴を記号の重みという形で抽出した．モデルのパラメータは，著者らの研究 [18] の公式を用いることで計算され，攻撃検出も単純な四則演算のみで行うことができる．そのため，提案モデルを用いて攻撃検出を行う場合でも，ModSecurity を用いて攻撃検出を行う場合でも，攻撃検出に要する処理時間の差はほとんど感じることはない．さらに文献 [18] では，データに欠損があるときに潜在曲線モデルはデータに対してあてはまりが良くなる例があることを指摘している．提案手法ではデータを2次関数にあてはめる方法を採用したため，用意したテストデータに対する検出性能を高めるために5つの記号にしか重みを付与することができなかった．モデルを改良することで入力可能なすべての記号に対して重みを付与す

ることができると考えられるため，モデルを改良することが今後の課題である．

7. まとめ

本研究では，入力文字列に含まれる記号に重みを付与し，その情報を用いて潜在曲線モデルを応用することで SQL インジェクション攻撃を検出する手法を提案し，SQL インジェクション攻撃を検出する Apache のモジュールを開発した．攻撃を検出する部分のモジュールの一部を著者の Web サイトで公開している．

http://matsudalab.office-server.co.jp/top/result/int01.html

検出実験の結果，提案手法では正常データに対する検出は高いパフォーマンスを示した一方，攻撃データに対するパフォーマンスは ModSecurity や SVM を用いた手法より高くないという結果が得られた．この結果は，学習段階においてどのようなデータを用いてモデルに学習させればよいかということを考える必要性があることを示しているものと考えられる．今後の課題は上述の学習データの構成方法の検討のほか，モデルを改良して正常データに対するパフォーマンスを下げずに，攻撃検出率を向上させることがあげられる．

参考文献

[1] OWASP, available from ⟨https://www.owasp.org/index.php/Main_Page⟩ (accessed 2014-11-28).
[2] Sadeghian, A., Zamani, M. and Ibrahim, S.: SQL Injection Is Still Alive: A Study on SQL Injection Signature Evasion Techniques, *2013 International Conference on Informatics and Creative Multimedia* (*ICICM*), pp.265–268 (2013).
[3] 小菅祐史，花岡美幸，河野健二：SQL インジェクション攻撃の脆弱性の効果的な自動検出手法，情報処理学会研究報告 CSEC，［コンピュータセキュリティ］，Vol.2008, No.45, pp.103–108 (2008).
[4] IPA：安全な SQL の呼び出し方，入手先 ⟨https://www.ipa.go.jp/security/vuln/press/201003_websecurity_sql.html⟩ （参照 2014-11-28）.
[5] Takahashi, H., Ahmad, H.F. and Mori, K.: Application for Autonomous Decentralized Multi Layer Cache System to Web Application Firewall, *2011 10th International Symposium on Autonomous Decentralized Systems* (*ISADS*), pp.113–120 (2011).
[6] Komiya, R., Incheon, P. and Hisada, M.: Classification of malicious Web code by machine learning, *2011 3rd International Conference on Awareness Science and Technology* (*iCAST*), pp.406–411 (2011).
[7] 伊波　靖，高良富夫：サポートベクタマシンを用いた WAF への異常検知機能の実装と評価，情報処理学会論文誌コンピューティングシステム，Vol.7, No.1, pp.1–13 (2014).
[8] Wang, Y. and Li, Z.: SQL Injection Detection with Composite Kernel in Support Vector Machine, *International Journal of Security & Its Applications*, Vol.6, No.2, pp.191–196 (2012).
[9] Scutum, available from ⟨http://www.scutum.jp/information/waf_tech_blog/⟩ (accessed 2014-11-28).
[10] 豊田秀樹：共分散構造分析［入門編］—構造方程式モデリング，朝倉書店 (1998).
[11] ModSecurity, available from ⟨https://www.modsecurity.org⟩ (accessed 2014-11-28).
[12] Matsuda, T., Koizumi, D., Sonoda, M. and Hirasawa, S.; On predictive errors of SQL injection attack detection by the feature of the single character, *2011 IEEE International Conference on Systems, Man, and Cybernetics* (*SMC*), pp.1722–1727 (2011).
[13] Matsuda, T.: Feature extraction of Web application attacks based on zeta distributions, *2013 World Congress on Internet Security* (*WorldCIS*), pp.119–121 (2013).
[14] 佐野綾子，松田　健，園田道夫，趙　晋輝：SQL インジェクション攻撃に含まれる文字の出現頻度とその関連性の解析による攻撃検出方法の提案，情報処理学会第 76 回全国大会講演論文集 (2014).
[15] Oosawa, T. and Matsuda, T.: SQL Injection Attack Detection Method using the Approximation Function of Zeta Distribution, *2014 IEEE International Conference on Systems, Man, and Cybernetics* (*SMC*), pp.833–838 (2014).
[16] Testing for SQL Injection (OTG-INPVAL-005), available from ⟨https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OTG-INPVAL-005)⟩ (accessed 2014-11-28).
[17] OWASP ModSecurity Core Rule Set, available from ⟨http://spiderlabs.github.io/owasp-modsecurity-crs/⟩ (accessed 2015-03-17).
[18] 松田　健：潜在曲線モデルを用いた能動的学習態度の推定，情報処理学会研究報告 MPS，数理モデル化と問題解決研究報告，Vol.2014-MPS-100, No.26, pp.1–4 (2014).

園田 道夫 （正会員）

1962 年生．1986 年明治学院大学英文学科卒業．学士（文学）．2007 年サイバー大学准教授．2014 年サイバー大学教授．情報セキュリティの研究に従事．IEEE 会員．

松田 健 （正会員）

1979 年生．2002 年東京理科大学理学部応用数学科卒業．2004 年同大学大学院修士課程修了．2010 年東京工業大学大学院博士課程修了．博士（理学）．2011 年サイバー大学講師．2013 年静岡理工科大学講師．情報数理の研究に従事．IEEE 会員．