L-033

構造方程式モデリングの応用による SQL インジェクション攻撃の検出と可視化技法

Visualization and Detection of SQL Injection Attack using Structural Equation Modeling

松田　健 †
Takeshi Matsuda

1. まえがき

我々が日常的に利用しているウェブ上のサービスは不正アクセスやアカウントの搾取など，様々なセキュリティ上の脅威にさらされている．このようなサイバー攻撃には多種多様な手法の存在が知られており，それらの対策法についても様々な手法が研究され，公開されている [1]．それにも関わらず，アプリケーションのセキュリティを向上させるガイドラインやツールを提供している OWASP（Open Web Application Security Project）[2] の情報によれば，サイバー攻撃による被害は増加の一途をたどっている．OWASP は攻撃者のターゲットになり易い攻撃を OWASP Top 10 として公表しており，SQL インジェクション攻撃はその中でも重大な損害をもたらす攻撃として紹介されている．

SQL インジェクション攻撃は，データベース駆動型のウェブアプリケーションの脆弱性を悪用し，データベースへ不正にアクセスする攻撃手法である．この攻撃は悪意ある SQL クエリをウェブアプリケーションへの入力データに挿入 (インジェクション) することで実現される．それゆえに，SQL インジェクション攻撃には特徴的な記号（以下，攻撃特徴記号という）が含まれることが多い [3], [4]．

SQL インジェクション攻撃に対する基本的な対策は，入力文字のエスケープ処理と，プリペアドステートメントを利用したアプリケーション開発である [1]．その他，ウェブアプリケーションに渡される文字列を解析して攻撃を検出する WAF（Web Application Firewall）を導入して防御する方法も提案されている [5], [6]．さらに，攻撃データと正常データを含む観測データに機械学習のアルゴリズムを適用し，学習用の観測データに含まれない未知の攻撃を検出する手法についても様々な研究がなされている [7], [8]．

アプリケーションに渡される文字列を解析して攻撃を検出する場合，第一種過誤（正常を攻撃として検出）と第二種過誤（攻撃を正常として検出）の問題が発生する．この問題については我々は，例えば [9]，[10] において，フリーの WAF で Apache のモジュールである ModSecurity では SQL インジェクション攻撃の検出には有効的であるが第一種過誤が起きやすい可能性があることを指摘し，構造方程式モデリングの手法を応用した攻撃検出法とデータの可視化技法を提案している．[9] で提案した攻撃検出手法を実装して開発した SQL インジェクション攻撃検出のための Apache モジュールは，著者のウェブページ [11] でその一部を公開している．しかしながら，第一種過誤も第二種過誤も引き起こさずに SQL インジェクション攻撃を検出することは困難であるため，精度の高い攻撃検出法の開発に加え，効率的にウェブサーバのログデータから攻撃の痕跡を発見する方法を確立して，攻撃が観測されたかどうかの判断を必要とする技術者を支援することは重要な問題であると考えられる．

そこで本研究では，[9]，[10] の手法を応用することでウェブサーバのログデータを可視化し，ログデータに含まれる SQL インジェクション攻撃の痕跡発見を支援する手法を提案する．本研究で使用するウェブサーバのログデータのうち，正常データについては実際に運用されているホームページから取得したデータを用意した．一方，攻撃データを含むログデータについては，用意した正常データのみからなるログデータに，実際に攻撃として使用可能なデータを人工的に付け加えることで生成した．本研究で使用した攻撃データは，URL に攻撃を挿入するタイプのものである．さらに，顔文字や Wiki 文法などの第一種過誤を引き起こしやすい正常データを用意し，正常なログデータにこれらのデータを挿入することで，ログデータがどのように可視化されるか考察を行った．本研究の実験の結果から，ログデータが攻撃データを一つでも含む場合，3 章で提案する攻撃特徴量 S の値に変化が見られるため，日常的に可視化されたログデータを確認することで SQL インジェクション攻撃の痕跡を迅速に発見できることが期待される．本論文の残りの章構成は以下の通りである．2 章では SQL インジェクション攻撃について紹介する．3 章では攻撃検出のためのアルゴリズムとログデータの可視化の方法について簡単にまとめ，4 章でデータを用いた実験を行う．最後に，5 章で考察とまとめを行う．

2.SQL インジェクション攻撃の概要と既存対策法

ウェブアプリケーションには，ユーザの入力からデータベースを検索するための SQL 文を組み立てる場面が存在する．SQL インジェクション攻撃は，このような場面において，開発者が想定していないユーザからの入力が行われたときに発生するウェブアプリケーション攻撃である．本研究では，URL に攻撃を挿入するタイプの攻撃データを用いて攻撃検出とログデータの可視化を行うため，これに関連する攻撃のサンプルを紹介し，SQL インジェクション攻撃のメカニズムについて説明する．

ここではユーザが自身の ID（777 とする）を入力し，そ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 文を直接生成するようなウェブアプリケーショ

†静岡理工科大学総合情報学部

ンを想定する．

http://www.sqlinjection.com?id=777

ウェブサーバはアプリケーションサーバに id=777 という情報を送り，データベースに送るための SQL 文を生成して id=777 に紐づいたユーザの情報をウェブページとして表示するものとする．ここで URL を

http://www.sqlinjection.com?id=777’ or ’XYZ ’ = ‘ XYZ

と変更すると，文字列 XYZ について XYZ=XYZ は常に真であるため，id=777 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ウェブページに表示される危険性をもつことになる．

SQL インジェクション攻撃による被害としてはデータベースの不正アクセスによるデータの破壊や改ざん，サーバの乗っ取りなどが挙げられる．このような攻撃からウェブアプリケーションを守るには，ユーザからの入力のチェックを行うことが重要となる．例えば，上で紹介した入力文字列

777’ or ’XYZ ’ = ‘ XYZ

では，シングルクォート（ ’ ）をダブルクォート（”）に置き換えるエスケープ処理を施したり，ユーザによって入力されるシングルクォートやセミコロンなどの特殊記号を単なる文字列として捉えるプリペアドステートメントをアプリケーションの開発に利用することで SQL インジェクション攻撃を防ぐことができる．このような根本的な攻撃対策技術が存在するにも関わらず，攻撃の被害が減少しない理由として，これらの対策が十分でないウェブアプリケーションが今なお存在することが挙げられる．また，仮に SQL インジェクション攻撃の対策が十分であったとしても，このような攻撃が実際に観測されていないかどうか定期的に確認して把握することも大切なことである．そこで本研究では，SQL インジェクション攻撃を自動検出する方法を考えるだけでなく，攻撃が観測されたかどうか容易に確認できるようにウェブサーバのログデータを可視化する方法について検討を行う．

3. 攻撃検出とデータ可視化アルゴリズム

SQL インジェクション攻撃には，悪意のある SQL 文を挿入するためにいくつかの特徴的な記号が含まれることが多い．そのような記号のことを著者らは攻撃特徴記号と呼び，SQL インジェクション攻撃のサンプルを収集 [12]・解析し，サンプルに含まれる攻撃特徴記号の分布（出現頻度の高い順に記号をソートする）がベキ乗則の一種であるゼータ分布に近い形になることを指摘している [4]．このようにして得られる統計情報は，SQL インジェクション攻撃によく使われる記号の種類はごく僅かであるが，攻撃の種類によっては様々な記号が稀に使用されることを示していると考えられる．SQL インジェクション攻撃のデータには特殊記号が含まれやすいため，このような攻撃特徴記号を用いた攻撃検出は有効的な手法であると考えられるが，攻撃特徴記号を含む正常データも容易に作ることができるため，第一種過誤が引き起こされる可能性が考えられる．第一種過誤を起こさないように攻撃特徴記号の使用方法を緩和すると今度は第二種過誤が引き起こされることになる．このような SQL インジェクション攻撃における第一種過誤と第二種過誤の問題に対処するために，構造方程式モデリングの手法に基づき，潜在変数の考え方を導入して攻撃を検出するだけでなく，データである文字列に攻撃と正常の特徴を色付けしてデータを可視化する方法を提案している [9], [10]．[10] では，画像を生成することで文字列に色付けしてデータを可視化する方法を提案している．しかしながら，ウェブサーバのログデータには大量の文字列が含まれるため，ログデータの文字列に直接的に色付けして可視化しても，その全体像を把握することは容易でない．そこで本研究では，[9], [10] で提案したアルゴリズムを攻撃検出に利用するだけでなく，3.2.2 で示す手法によりウェブサーバのログデータを可視化する方法を提案する．

3.1. 攻撃検出アルゴリズム

3.1.1. 特徴空間の構成

攻撃データを I_A 個，正常データを I_N 個準備し，攻撃データ集合と正常データ集合をそれぞれ作る．攻撃データ集合に含まれるすべての記号の出現頻度を計算し，記号を出現頻度の大きい順に並べ替え，

$$s_1, s_2, \cdots, s_J$$

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

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

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

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

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 $(x =)5$ 個の記号を攻撃特徴記号として攻撃検出やログデータの可視化に利用する．以上の準備のもとで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．

3.1.2. 攻撃検出アルゴリズム

3.1.1 で構成した特徴空間に以下の数理モデルを当てはめることを考える．

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

ただし，ϵ_i（$i = 1, 2, 3, 4$）は平均 0，分散 $\sigma_i^2 > 0$ の正規分布にしたがう誤差項であるとする．まず各データ l_i に対して，尤度関数

$$\prod_{x=1}^{5} \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp \frac{(y_x - a_0 + a_1 x + a_2 x^2)^2}{2\sigma_1^2}$$

を最大化するパラメータ $a_0^{(i)}, a_1^{(i)}, a_2^{(i)}$ を計算し，これと同様の方法で $a_0^{(i)}, a_1^{(i)}, a_2^{(i)}$ と t の値からパラメータ

$$b_{00}, b_{10}, b_{01}, b_{11}, b_{02}, b_{12}$$

を計算する．具体的な計算方法は [13] で与えられている．

以下は，[10] で使用した学習用データ（攻撃データ：2779 個，正常データ：444 個）による計算結果である．

$$\begin{aligned} y &= (b_{00} + b_{10} t) + (b_{01} + b_{11} t) x + (b_{02} + b_{12} t) x^2 \\ b_{00} &= 0.367642642642704 \\ b_{10} &= 0.520581649276325 \\ b_{01} &= 0.0093468468468900 \\ b_{11} &= -0.427524704823484 \\ b_{02} &= -0.00379129129129632 \\ b_{12} &= 0.061190887111532 \end{aligned}$$

ここで，曲線 $y_x = a_0 + a_1 x + a_2 x^2$ において

$$\begin{aligned} a_0 &= b_{00} + b_{10} t + \epsilon_2 \\ a_1 &= b_{01} + b_{11} t + \epsilon_3 \\ a_2 &= b_{02} + b_{12} t + \epsilon_4 \end{aligned}$$

と定義し，$t = 1$ を攻撃に対応させてことから，これらの値を $y_x = a_0 + a_1 x + a_2 x^2$ に代入することで

$$y = 0.8882 - 0.4182x + 0.0574x^2 \qquad (1)$$

という曲線が特徴空間上で SQL インジェクション攻撃データの特徴を表しているものと考える．同様に $t = 0$ を代入し，

$$y = 0.3676 + 0.0093x - 0.038x^2 \qquad (2)$$

という曲線が特徴空間上で正常データの特徴を表しているものと考える．曲線の係数は小数第 5 位で四捨五入したものである．

表 1: 横軸 (x 軸) に対応する 5 つの記号

記号	Space	’	;	)	(
x 座標	1	2	3	4	5

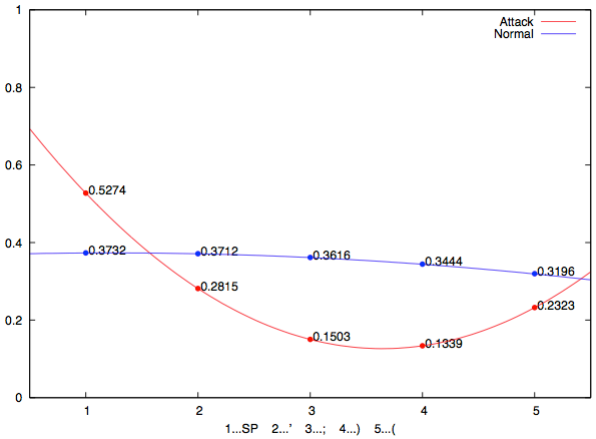

図 1: 攻撃と正常の特徴を表現する曲線

攻撃検出は，与えられた入力の特徴空間上の座標を計算し，攻撃と正常のどちらの曲線に当てはまりが良いか残差を計算することで行う．ただし，与えられた入力には表 1 にある 5 つの攻撃特徴記号がすべて含まれるとは限らないため，該当する記号が入力に存在しない場合は，その記号に関する残差の計算は行わないものとする．

3.2. 攻撃の可視化

ここでは，3.1 で紹介した手法を応用して攻撃データを可視化する手法を提案する．

3.2.1. 既存手法

まず [10] で提案した攻撃データの可視化手法とその結果を紹介する．式 (1) と (2) を用いて，$0 \leq y \leq 1$ に色の三原色である RGB 値 $[0, 255]$ を対応させ，正常の特徴には青 (B 値) を，攻撃の特徴には赤 (R 値) を表 2 のように対応させ，データに着色することで可視化したものが図 2（攻撃），図 3（正常）である．

図 2 は攻撃データを一行毎に並べたものであり，全体的にデータが赤く着色されていることが分かる．一方，図 3 は正常データを一行毎に並べたものであり，図 2 と比較すると赤い部分が少ないことがわかる．

表 2: 特徴の着色

記号	正常特徴	攻撃特徴
Space	95.165541	134.498738
’	94.648649	71.774075
;	92.198198	38.323205
)	87.814189	34.146130
(	81.496622	59.242849

図 2: 攻撃データへの着色

図 3: 正常データへの着色

3.2.2. 提案手法

本研究では，ウェブサーバのログデータを可視化する方法について考える．3.2.1 の方法をログデータにそのまま適用すると，ログデータ自体に攻撃特徴記号が含まれている場合も考えられるうえ，データの量も少なくないためデータに色をつけるだけでは実用的でないと考えられる．そこで本研究では，ログデータにどれくらいの割合で攻撃特徴記号が含まれるているかという情報を図 4 のようなインターフェースを作成して可視化する手法を提案する．棒グラフの赤色（網掛け）の部分で攻撃特徴の割合を表している．なお，図 4 に示したログデータには SQL インジェクション攻撃のデータは含まれていない．攻撃特徴の割合は表 2 の攻撃特

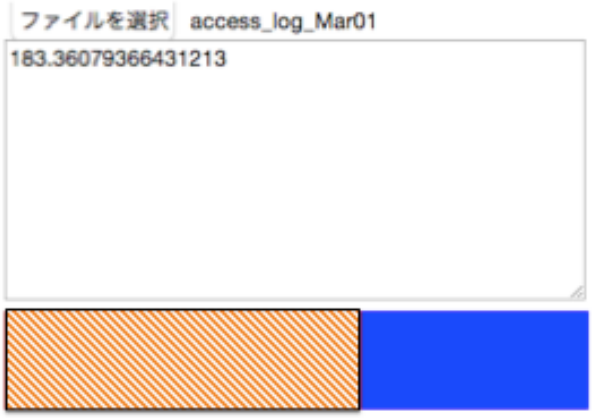

図 4: ログデータの可視化

徴の値を用い，以下の式で計算した．

$$S = \frac{\alpha_1 z_1 + \alpha_2 z_2 + \alpha_3 z_3 + \alpha_4 z_4 + \alpha_5 z_5}{\alpha_1 z_1 + \alpha_2 z_2 + \alpha_3 z_3 + \alpha_4 z_4 + \alpha_5 z_5 + z_6} \quad (3)$$

ただし，

$$\begin{aligned} \alpha_1 &= 95.165541 \\ \alpha_2 &= 94.648649 \\ \alpha_3 &= 92.198198 \\ \alpha_4 &= 87.814189 \\ \alpha_5 &= 81.496622 \end{aligned}$$

であり，z_1, z_2, z_3, z_4, z_5 はそれぞれログデータに含まれる Space，シングルクォート，セミコロン，右丸括弧，左丸括弧の出現頻度である．z_6 は，ログデータに含まれるその他の記号や文字，数字などの出現頻度である．$\alpha_i = 1$ としたときはログデータにおける上記 5 つの記号の含有率であり，$\alpha_i > 1$ とすると S の値は含有率より大きくなることは簡単に示すことができる．実際，

$$\frac{\sum_{j=1}^{J} \alpha_j s_j}{\left(\sum_{j=1}^{J} \alpha_j s_j\right) + s_{J+1}} - \frac{\sum_{j=1}^{J} s_j}{\sum_{j=1}^{J+1} s_j}$$

を計算すると，分子において

$$\begin{aligned} &\left(\sum_{j=1}^{J} \alpha_j s_j\right)\left(s_{J+1} + \sum_{j=1}^{J} s_j\right) - \left(\sum_{j=1}^{J} s_j\right)\left(s_{J+1} + \sum_{j=1}^{J} \alpha_j s_j\right) \\ &= \left(\sum_{j=1}^{J} \alpha_j s_j\right)\left(\sum_{j=1}^{J} s_j\right) + \left(\sum_{j=1}^{J} \alpha_j s_j\right) s_{J+1} \\ &\quad - \left(\sum_{j=1}^{J} s_j\right)\left(\sum_{j=1}^{J} \alpha_j s_j\right) - \left(\sum_{j=1}^{J} s_j\right) s_{J+1} \end{aligned}$$

$$= \left(\sum_{j=1}^{J} (\alpha_j - 1) s_j \right) s_{J+1}$$

が成り立つ．

次章では，本研究の提案手法を実際のウェブサーバのログデータに適用した結果を紹介する．ただし，攻撃データについては，ローカル環境において実際に SQL インジェクション攻撃を引き起こすデータを入力して攻撃検出を行い，ログデータを生成した．

4. ログデータ可視化の実験

前章で提案したログデータの可視化手法を，実際のウェブサーバのログデータに適用し，そのデータにローカル環境で SQL インジェクション攻撃を仕掛けて生成したログデータを加えて，ログデータがどのように可視化されるか実験を行った．本論文で用いたログデータは以下の通りである．

(実験 1) 一週間分のログデータ (攻撃は含まない)
(実験 2) ある一日のログデータに一つだけ攻撃データを挿入したログデータ
(実験 3) ある一日のログデータに二つだけ攻撃データを挿入したログデータ
(実験 4) ある一日のログデータに五つだけ攻撃データを挿入したログデータ
(実験 5) ある一日のログデータに第一種過誤を起こし易い正常データを挿入したログデータ

なお，実験にはパラメータの学習用に用いたものとは別に用意したデータ (攻撃 : 200 個，正常 : 50 個) をランダムに選択して使用した．このデータに 3.1.2 の攻撃検出手法を適用した結果とその考察については [9] にあるため，ここではその結果のみ表 3 に示す．

表 3: 検出実験の結果

	攻撃	正常
学習用データ	2779 個	444 個
テスト用データ	200 個	50 個
検出率	96%	100%

表 3 の結果から，3.1.2 の攻撃検出手法 [9] は顔文字のような記号を多く含む正常データについて第一種過誤を引き起こしにくいことがわかる．例えば，

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

は正常データであり，3.1.2 の攻撃検出手法では正常として検出するが，Apache のフリーのモジュールである ModSecurity と呼ばれる WAF をデフォルトの状態で使用して上記の正常データを入力すると，ModSecurity は攻撃として検出する．

次に，本研究の提案手法を用いて（実験 1）と（実験 5）のデータを可視化した結果を図 5 に示す．棒グラ

図 5: （実験 1）と（実験 5）の比較

フの網掛けの部分が攻撃特徴量 S を示している．ログデータには大量の記号が含まれているため，攻撃データを含まないログデータであっても網掛け部分の領域が多くなっている．しかしながら，提案手法を用いてログデータを可視化した場合，顔文字のような記号を多く含む正常データがログデータに含まれている場合でも，網掛け部分の領域はほとんど広がらないことが図 5 からわかる．

次に，(実験 1)，(実験 2)，(実験 3)，(実験 4) のログデータを可視化したものの比較を図 6 に示す．本研

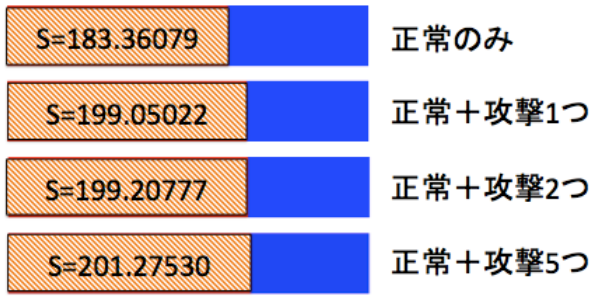

図 6: （実験 1）から（実験 4）までの比較

究手法を用いて可視化した場合，ログデータに一つでも攻撃データが含まれると，多少ではあるが，棒グラフの網掛け部分の領域が広がることが図 6 からわかる．

最後に，攻撃データを含まない連続する一週間分のログデータを可視化したものの比較を図 7 に示す．日によって多少のばらつきが見られるものの，データを取得したウェブサーバでは攻撃特徴量 S が 200 近くになる場合に攻撃データを含むということが図 6 と図 7 から読み取ることができる．ただし，ウェブサーバのコンテンツやアクセス数によって S の値は変化するため，他の環境化ではログデータに攻撃が含まれるかどうかの基準値は環境によって異なる値をとることに注意しなければならない．

5. 考察とまとめ

本研究では，構造方程式モデリングの一つの手法である潜在曲線モデルの考え方を応用し，ウェブサーバのログデータを可視化する手法を提案した．攻撃検出については，提案手法と ModSecurity や機械学習を用

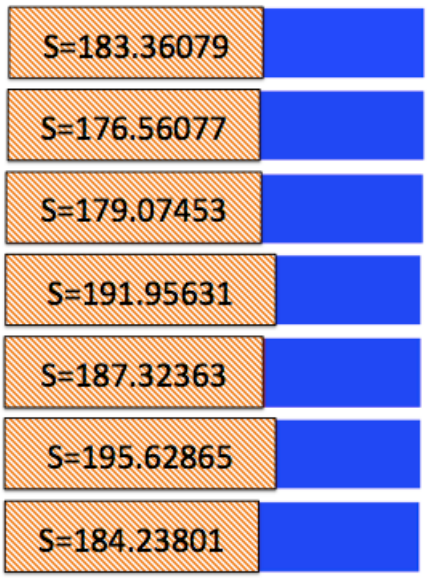

図 7: 一週間分の正常ログデータの比較

いた場合との比較を [9] で行っている．本研究のように，入力データに含まれた記号に着目して SQL インジェクション攻撃を行う場合は，記号を多く含む正常な入力データを攻撃として検出しないように注意しなければならないが，前章の実験の結果から本研究の提案手法はこのような第一種過誤を引き起こしにくいことが図 5 の可視化した結果からも読み取ることができる．しかしながら，ModSecurity は第一種過誤を引き起こしやすい傾向にあるものの，かなりの精度で SQL インジェクション攻撃を検出することができるため，第一種過誤を引き起こさずに攻撃検出の精度をさらに高めることは今後の重要な課題である．また，本研究では一つのウェブサーバから取得したログデータのみを使用して実験を行ったため，他のウェブサーバのログデータに本研究の手法を適用してどのようにログデータが可視化されるか調べること，攻撃特徴量を計算するモデルを精密化し，可視化するグラフをさらに見やすく改良することも今後の課題である．

謝辞

攻撃検出システム開発には合同会社 binary lab の大谷康介氏に協力頂いたことを感謝する．

以上

参考文献

[1] IPA, “安全なウェブサイトの作り方,” https://www.ipa.go.jp/files/000017316.pdf (2015 年 4 月 12 日確認)

[2] The Open Web Application Security Project (OWASP), https://www.owasp.org/index.php/Main_Page (2015 年 4 月 12 日確認)

[3] T. Matsuda, D. Koizumi, M. Sonoda and S. Hirasawa, “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.

[4] T. Matsuda, “Feature extraction of Web application attacks based on zeta distributions,” 2013 World Congress on Internet Security (WorldCIS), pp.119-121, 2013.

[5] H. Takahashi, H. F .Ahmad and K. Mori, “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] ModSecurity，https://www.modsecurity.org (2015 年 4 月 12 日確認)

[7] Yi Wang and Zhoujun Li, “SQL Injection Detection with Composite Kernel in Support Vector Machine,” 2011 10th International Journal of Security & Its Applications, Vol. 6 Issue 2, pp.191-196, 2012.

[8] E. H. Cheon, Z. Huang and Y. S. Lee, “Preventing SQL Injection Attack Based on Machine Learning,” International Journal of Advancements in Computing Technology(IJACT) Volume5, Number9, May 2013.

[9] 園田道夫, 松田　健, “攻撃特徴記号に基づく WAF 開発,” 情報処理学会論文誌, CSEC 特集号, （Submitted）

[10] 藤岡あやか, 松田　健, 園田道夫, 趙　晋輝, “潜在曲線を用いた着色による SQL インジェクション攻撃の特徴の可視化,” 情報処理学会 第 77 回全国大会講演論文集, 2015

[11] “構造方程式モデリングを応用した攻撃検出法,”http://matsudalab.office-server.co.jp/top/result/int01.html (2015 年 4 月 12 日確認)

[12] “Testing for SQL Injection (OTG-INPVAL-005),” https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OTG−INPVAL-005) (2015 年 4 月 12 日確認)

[13] 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 2014-MPS-100(26), pp.1-4 2014.