SQL インジェクション攻撃の自動検出アルゴリズムとその数理モデルに関する考察

松田　健

要　旨

SQL インジェクション攻撃は，データベースをもつ Web アプリケーションにとって深刻な脅威となっている。この攻撃から Web アプリケーションを防御する方法として構文解析やパターン認識を用いた攻撃検出法が開発されているが，今なお新しい攻撃が開発され続けておりこれらの方法だけで対処することができないのが実情である。この問題に対して，過去に観測された攻撃をブラックリスト化して攻撃検出を行う Web アプリケーションファイアウォールを導入する方法が一般的になっているが，ブラックリストの肥大化やそれに伴って検出に時間がかかることなどが問題となっている。本研究では，ブラックリストに依存せず，未知の攻撃も検出可能な SQL インジェクション攻撃の自動検出を支援するための数理モデルを提案した。さらに，数理モデルの性能を測るための予測誤差を定義し，2 値判別の問題で広く利用されているシグモイド関数を用いた数理モデルと提案モデルの予測誤差を比較することで，提案モデルの有効性を示した。

＊本論文は 2011 IEEE 国際会議に採択された論文の内容を詳細に書き換えたものである。

キーワード　SQL インジェクション攻撃，検出，数理モデル

1. はじめに

SQL インジェクションとは Web アプリケーションがもつ脆弱性の一種であり，これを悪用した攻撃は SQL インジェクション攻撃と呼ばれている。近年，ネットショッピングやインターネットバンクなどオンライン上での取引が盛んになり，個人情報を含むデータベースをもった商用の Web アプリケーションが広く導入されており，SQL インジェクション攻撃はこのような Web アプリケーションにとって深刻な脅威となっている。SQL インジェクション攻撃は Web ページのフォームを介して行う比較的単純な攻撃であり，Web アプリケーションのデータベースへ不正にアクセスし，データベースの情報を閲覧したり，改ざんしたりすることを目的として行われる。

SQL インジェクション攻撃は 1999 年に電子雑誌 Phrack [(1)] でその危険性についてはじめて報告され，2005 年頃から多くの攻撃が観測されるようになった。さらに 2007 年頃

サイバー大学 IT 総合学部・講師
原稿受付日：2011 年 10 月 3 日
原稿受理日：2011 年 12 月 13 日

から自動拡散するワームやボットネットを利用した攻撃が盛んとなり，現在でも多くの攻撃が観測されている[2]。

SQL インジェクション攻撃から Web アプリケーションを防御する有効な技術的方法は既にいくつか開発されている。もっとも単純なものは入力を制限する方法であり，攻撃を無害化するサニタイズと呼ばれる方法が利用されている[3]。その他にも，SQL インジェクション攻撃の文字列の構文解析や，パターン認識を行う手法も提案されている[4][5][6][7][8]。しかしながら，これらの手法の有用性は認められているにも関わらず，現在でも SQL インジェクション攻撃による被害が多数報告されている。その原因として次の 3 つの問題が挙げられる。

- 入力制限やサニタイズを Web アプリケーションに実装する際にヒューマンエラーが発生する可能性があること
- 既に導入され稼動しているすべての Web アプリケーションに対してメンテナンスを行うことが現実的でないこと
- 構文解析やパターン認識を応用した攻撃検出では，それらを回避する攻撃が開発されるため将来的に攻撃検出ができなくなること

そのために現状では，SQL インジェクション攻撃から Web アプリケーションを防御するための方法として，Web アプリケーションファイアウォール（WAF）が広く利用されている。WAF は過去に観測された SQL インジェクション攻撃をブラックリスト化し，Web ページから入力された文字列とブラックリストを照合して攻撃検出を行う。したがって，攻撃を少し改変し亜種を作成するだけで攻撃検出ができなくなること，新しい攻撃が観測される度にブラックリストを更新しなければならないためブラックリストが肥大化するだけでなく，検出にかかる時間もそれに伴って多くなることが問題である。

これらの問題に対し，本論文では，文字単位を特徴とする SQL インジェクション攻撃検出アルゴリズムと検出を行うための数理モデルを提案する[9][10]。提案モデルは，フォームに入力された文字列が SQL インジェクション攻撃であるかどうかの判別を行う 2 値判別を行うものである。2 値判別の問題ではシグモイド関数を応用した数理モデルが広く利用されており[11][12][13]，SQL インジェクション攻撃の検出にも利用することができる。そこで，我々は数理モデルの予測性能を評価するための予測誤差を定義し，提案モデルとシグモイド関数を応用したモデルの予測誤差を比較することで提案モデルの有効性を示した。

本研究で提案する SQL インジェクション攻撃検出を行う数理モデルは，攻撃文字列と正常文字列の両方の検出率を計算し，その両方の検出率を高くするものであり，Web ページから入力された文字列に含まれるいくつかの特定の文字や記号の含有率によって入力文字列が SQL インジェクション攻撃であるかどうかを判別する。そのため，WAF のようにブラックリストの更新を必要としない点に特徴をもっている。SQL インジェクション攻撃のパターンが変わった場合にでも，攻撃検出に利用する最適な文字を選ぶことによっ

て対処可能なところも我々の提案モデルの特徴である。

本論文の構成は以下の通りである。2 章で，SQL インジェクション攻撃の実例を用いて我々の提案アルゴリズムの紹介し，3 章で，SQL インジェクション攻撃検出のための数理モデルの紹介を行い，提案アルゴリズムとモデルに対する考察を行う。最後の 4 章でまとめを行い，今後の課題を示す。

2. SQL インジェクション攻撃とその検出アルゴリズム

この章では，SQL インジェクション攻撃について概説し，SQL インジェクション攻撃を検出するための我々の提案モデルの紹介を行う。

2.1 SQL インジェクション攻撃

SQL インジェクション攻撃とは，攻撃者が Web ページのフォームを通して悪意のある SQL クエリを SQL 文に挿入する攻撃である。攻撃が成功すれば，攻撃者はデータベースに不正にアクセスすることができ，情報を改竄したり消去したりすることができる。

SQL インジェクション攻撃にはいくつかのパターンがあり，それらは攻撃者の目的に依存している。SQL インジェクション攻撃は，文献 [14] [15] の中で以下の 7 つのパターンに分類されている。

- Tautologies
  (トートロジー：常に真となる命題)
- Illegal/Logically Incorrect Queries
  (不正な / 論理的に正しくない問い合わせ)
- Union Query
  (同じ構造をもつ複数のテーブルやクエリを 1 つの結果として表示)
- Piggy-backed Queries
  (ピギーバック，貨物輸送の形態)
- Stored Procedure
  (1 つ以上の手続きを定義したもの)
- Inference
  (システムのエラー情報をもとにした攻撃)
- Alternate Encoding
  (文字列をエンコード化する攻撃)

本章では，このうち Tautologies と Piggy-backed Queries タイプの攻撃を実例としてとりあげ，我々の提案手法に適用する。以下，Web ページのフォームの入力を想定し入力された文字列 (以下，入力文字列) が SQL インジェクション攻撃である場合それを攻撃

文字列と呼び，SQL インジェクション攻撃でない場合は正常文字列と呼ぶ。

例 2.1（Tautologies）

入力フォームに文字列が入力され

SELECT * FROM CardInfo WHERE ID=’name’and PASS=’777’OR‘7 = 7”

という SQL クエリが生成された場合を考える。このとき，この入力文字列の最後の部分の恒等式 7 = 7 という常に真となる命題があるため，入力時に無害化や入力制限などの処理が適切にされない場合，データベースは攻撃者に対してすべての情報のアクセスを許してしまう。

例 2.2（Piggy-backed Queries）

ここでは，ユーザー ID とパスワードを入力するフォームを想定した例を紹介する。パスワードの入力フォームに

7 ;drop table cardinfo

という文字列が入力されたとする。このとき，Web アプリケーションは以下の SQL クエリを生成する。

SELECT * FROM CardInfo WHERE ID=’name’pass= 7 ; DROP TABLE CardInfo

この攻撃では区切り文字のセミコロン（;）が使われているところに特徴があり，区切り文字の後に悪意のある SQL クエリが余分に付け加えられている。区切り文字セミコロン（;）に対して脆弱性をもつ Web アプリケーションは，セミコロン（;）が入力されていることから，元々の SQL クエリだけでなく付け加えられた悪意のある SQL クエリも通してしまう。その結果，この攻撃が成功すると，データベースから CardInfo という名前のテーブルが消去される。

2.2 SQL インジェクション攻撃検出アルゴリズム

本節では，文献 [10] で提案した SQL インジェクション攻撃の検出アルゴリズムについて紹介する。

まず，本論文で使用する記号の定義を行う。フォームから入力される入力文字列を l，記号・数字・アルファベットなどの単一文字を s，いくつかの単一文字たちの組み合わせからなる集合を S とする。我々の提案方法は，SQL インジェクション攻撃に含まれる単一の文字を攻撃特徴とするものである。文献 [9] [10] では，表 2.1 に示す 20 個の文字を利用して SQL インジェクション攻撃の特徴を良く捉えている文字の選別を行った。以下，SQL インジェクション攻撃検出を行うための準備を行う。

入力文字列 l に対して，

$$x = \frac{\#S}{|l|}$$

を定義し，0 から 1 の有理数に値をとる x を入力として扱う。ここで，$|l|$ は入力文字列長を，

$\#S$ は入力文字列 l に含まれる単一文字たちの総数を表すものとする。さらに，入力文字列全体の集合を L，攻撃文字列全体の集合を L_A，正常文字列全体の集合を L_N とすると

$$L = \{ l_i \mid i \in N \} = L_A \cup L_N, L_A \cap L_N = \phi$$

と表すことができる。ここで，N は自然数全体の集合を表す。SQL インジェクション攻撃の検出では，入力文字列が攻撃文字列であるか正常文字列であるかを判別する 2 値判別の問題を考えることが目標となる。つまり，すべての入力 $l_i \in L$ に対して $l_i \in L_A$ であるか，$l_i \in L_N$ であるかを判別する。そのために以下の判別関数を定義する。

$$h(x) = \begin{cases} 1 & (x \geq \alpha) \\ 0 & (x < \alpha) \end{cases}$$

ここで，α は 0 から 1 の値をとる実数で閾値と呼ばれる。$h(x)=1$ なら $l_i \in L_A$（攻撃文字列と判別）であり，$h(x)=0$ なら $l_i \in L_N$（正常文字列と判別）である。

表 2.1　攻撃を特徴づける文字の候補

変数	文字	変数	文字
s_i	半角スペース	s_{xi}	パーセント
s_{ii}	セミコロン	s_{xii}	ダブルクォーテーション
s_{iii}	シングルクォーテーション	s_{xiii}	アンバサンド
s_{iv}	右側丸括弧	s_{xiv}	バックスラッシュ
s_v	左側丸括弧	s_{xv}	パイプ
s_{vi}	右側中括弧	s_{xvi}	等号
s_{vii}	左側中括弧	s_{xvii}	大なり不等号
s_{viii}	右側大括弧	s_{xviii}	小なり不等号
s_{ix}	左側大括弧	s_{xix}	アスタリスク
s_x	シャープ	s_{xx}	スラッシュ

例 2.1 と例 2.2 で紹介した SQL インジェクション攻撃を用いて，判別関数の計算例を以下に示す。

例 2.3

l_1 を

SELECT * FROM CardInfo WHERE ID=’name’and PASS=’777’OR‘7 = 7”

とする。この攻撃を集合 $S = \{s_i, s_{ii}, s_{iii}\}$ で検出する場合の計算例を示す。集合 S には，半角スペース，セミコロン，シングルクォーテーションが含まれる。l_1 の文字列長は 59 であるから $|l_1| = 59$ となり，l_1 には半角スペース，セミコロン，シングルクォーテーションが合計 16 個含まれていることから $\#S = 16$ となることが分かる。したがって，

$$x = \frac{16}{59} \approx 0.2712$$，（≈は，およその数を表す記号とする）

であり，閾値を例えば$\alpha = 0.1$と設定しておけば判別関数は

$$h(x) = \begin{cases} 1 & (x \geq 0.1) \\ 0 & (x < 0.1) \end{cases}$$

となることから，$h\left(\frac{16}{59}\right) = 1$を得る。したがって，判別関数は入力文字列$l_1$を攻撃であると判別する。この入力文字列の場合は，閾値を$0 < \alpha \leq \frac{16}{59}$の範囲で設定しておけば入力文字列$l_1$に対する攻撃検出が可能である。

例 2.4

l_2を

SELECT * FROM CardInfo WHERE ID=’name’ pass= 7 ; DROP TABLE CardInfo

とする。この攻撃を上の例 2.3 と同様に集合$S = \{s_i, s_{ii}, s_{iii}\}$を用いた場合の計算例を示す。この場合は,

$$x = \frac{13}{66} \approx 0.1967$$

となり，閾値を例えば$\alpha = 0.1$と設定しておけば判別関数は

$$h(x) = \begin{cases} 1 & (x \geq 0.1) \\ 0 & (x < 0.1) \end{cases}$$

となることから，$h\left(\frac{13}{66}\right) = 1$を得る。したがって，判別関数は入力文字列$l_2$を攻撃であると判別する。この入力文字列の場合は，閾値を$0 < \alpha \leq \frac{13}{66}$の範囲で設定しておけば入力文字列$l_2$に対する攻撃検出が可能である。

以上の例から攻撃の検出には文字集合Sと閾値αの定め方が重要であることが分かる。文献[9][10]では以下に示すように，攻撃文字列を正常文字列と誤判別したり，正常文字列を攻撃文字列と誤判別したりする確率を小さくする最適な文字集合や閾値を選ぶためのアルゴリズムを提案した。

攻撃検出に最適な文字集合や閾値は，それらを定めるために利用する攻撃文字列と正常文字列のサンプルに依存する。したがって，新しい攻撃が観測されれば再び文字集合や閾値を定める必要がある。この点については Web アプリケーションファイアウォールのブラックリストと同様であるが，ブラックリストが肥大化したり，攻撃検出にかかる時間が増加したりすることは我々の提案方法では起こらないことが特徴の 1 つである。

文字集合 S と閾値 α を定めるために利用するサンプルのうち，n 個の攻撃文字列のサンプルを D_A，m 個の正常文字列のサンプルを D_N とする。k を自然数とし，すべての $l_i \in D_A \cup D_N$ に対して

$$x_{ik} = \frac{\#S_k}{|l_i|}$$

を計算する。さらに J 個の閾値 $\alpha_j (j = 1,2,\cdots,J)$ を用意し，

$$T_{kj} = \frac{\sum_{i=1}^{n} h(x_{ik})}{n}$$

$$F_{kj} = 1 - \frac{\sum_{i=1}^{m} h(x_{ik})}{m}$$

を計算する。ここで，T_{kj} は攻撃文字列の中で $h(x_{ik}) \geq \alpha_j$ を満足する割合を示すものであり，T_{kj} の値が 1 に近いほど攻撃文字列の検出率が高くなることを表す。同様に，F_{kj} の値が 1 に近いほど正常文字列を攻撃文字列であると誤検出率が低くなることを表す。T_{kj} と F_{kj} を R 回繰り返し計算することを T_{kjR} と F_{kjR} で表し，R 回中の r 回目の計算結果を T_{kjr} と F_{kjr} で表す。

［文字集合と閾値を定めるアルゴリズム］

（1） J 個の閾値候補 $\alpha_j (j = 1,2,\cdots,J)$ を固定する
（2） すべての i, j, S_k に対し T_{kj}, F_{kj} を計算し，これを R 回繰り返す
（3） すべての S_k に対し $\frac{T_R}{R} \geq 0.9$ を満足する閾値 α_j を選んでできる集合 A_k とする
（4） A_k のうち $\max\left\{(1-\beta)\frac{T_R}{R} + \beta\frac{F_R}{R}\right\}$ となる閾値 α_j を選ぶ

ただし，β は $0 \leq \beta \leq 1$ を満たす実数とし，$T_R = \sum_{r=1}^{R} T_{kjr}, F_R = \sum_{r=1}^{R} F_{kjr}$ とおいた。

β は加重平均を決めるための重みであり，β の値を 0 近づければ近づけるほど攻撃検出率を重視することになる。文献[9][10]で用いたサンプルの場合，文字集合は $S =$ {スペース，セミコロン，右側丸括弧} が，閾値は $\alpha = 0.08$ が選ばれ，さらに，いかなる β を選んでも $(1-\beta)\frac{T_R}{R} + \beta\frac{F_R}{R}$ は安定的に高くなるという結果が得られている。

3. SQL インジェクション攻撃検出モデル

前章では，SQL インジェクション攻撃を検出するためのアルゴリズムについて紹介した。本章では，我々の提案モデルに基づく SQL インジェクション攻撃の検出を行うための数理モデルを提案する。

以下，入力を x とし，出力を y とする。前章で紹介したアルゴリズムでは判別関数の値 $h(x)$ が出力 y に対応する。入力 x に対して出力が $y=1$ となった場合 x は攻撃文字列であり，$y=0$ となった場合 x は正常文字列であるとする。我々は文献 [9] で以下のような数理モデルを提案した。

$$p_1(y \mid x,b)=(x^b)^y(1-x^b)^{1-y}$$

数理モデル $p_1(y \mid x,b)$ は，入力 x が攻撃文字列である確率が x^b であり，正常文字列である確率が $1-x^b$ であることを表すベルヌーイ分布になっている。ただし，関数のパラメータ b は $0<b<1$ を満足する実数であり，関数 x^b は定義域，値域ともに実数上の閉区間 $[0,1]$ である。2 値判別の問題を考えるとき，

$$s(x)=\frac{1}{1+e^{-bx}}$$

で定義されるシグモイド関数が広く利用されている。SQL インジェクション攻撃の検出についてもシグモイド関数を適用することが可能であるが，シグモイド関数の定義域は実数全体であり，値域が $0 \le s(x) \le 1$ であることから，このままでは我々の提案アルゴリズムに基づいた攻撃検出モデルを作ることが出来ない。そこで，実数上の閉区間 $[0,1]$ から実数全体への連続関数

$$t=\begin{cases} -\infty, & (x=0) \\ \log\dfrac{x}{1-x}, & (x \in (0,1)) \\ +\infty, & (x=1) \end{cases}$$

を定義し，本論文では，これと関数 $s(t)$ との合成によってできる関数

$$g(x)=\frac{x^b}{x^b+(1-x^b)}$$

をシグモイド関数として扱う。これを用いた SQL インジェクション攻撃を検出するための数理モデルは

$$p_2(y \mid x,b)=\left(\frac{x^b}{x^b+(1-x^b)}\right)^y\left(1-\frac{x^b}{x^b+(1-x^b)}\right)^{1-y} \quad \cdots (2)$$

で与えられる。この場合，入力 x が攻撃文字列である確率は $\dfrac{x^b}{x^b+(1-x^b)}$ であり，正常文字列である確率は $1-\dfrac{x^b}{x^b+(1-x^b)}$ である。文献 [9] では，我々の提案モデルとシグモイド関数を用いたモデルとの検出の違いを測るための予測誤差を定義し，提案モデルの予測誤差がシグモイド関数を用いた場合のモデルの予測誤差と比べて小さくなることを理論的に

述べている。本論文では，文献 [9] で与えられている証明は紹介せずにその結果のみを述べ，考察を行う。

十分大きな自然数 K に対して，閉区間 $\left[0,\frac{1}{2}\right]$ の分割

$$T=\cup_{k=1}^{K}T_k=\cup_{k=1}^{K}\left[t_k,t_{k+1}\right]$$

を考える。このとき，

$$T'=\cup_{k=1}^{K}T'_k=\cup_{k=1}^{K}\left[1-t_{k+1},1-t_k\right]$$

は閉区間 $\left[\frac{1}{2},1\right]$ の分割となる。$p(x)$ で入力 x の分布を表し，$p(dx)$ で確率測度を表す。

このとき，以下に示す定理が成り立つ。

定理 3.1 ［文献 [9]］

次の 3 つの条件

1. $0<b<1$

2. $0\le\alpha\le\frac{1}{2}$

3. すべての k について，$p(T_k)\le p(T'_k)$

を満足するとき

$$e(p_2)>e(p_1)$$

が成り立つ。ただし，

$$e(p)=\Pr(h(x)\neq y)$$

である。

ここで，$\Pr(h(x)\neq y)$ は数理モデル p が誤判別する確率に対する期待値を表すものである。したがって，定理の不等式 $e(p_2)>e(p_1)$ は与えられた条件の下では，常にシグモイド関数を用いたモデルより我々の提案モデルのほうが誤判別は起きにくいことを表していることが分かる。

提案モデルでは入力 x が攻撃文字列である確率を x^b で表している。関数 x^b は図 3.1 から分かるようにパラメータのとり方でグラフの形状が異なる。

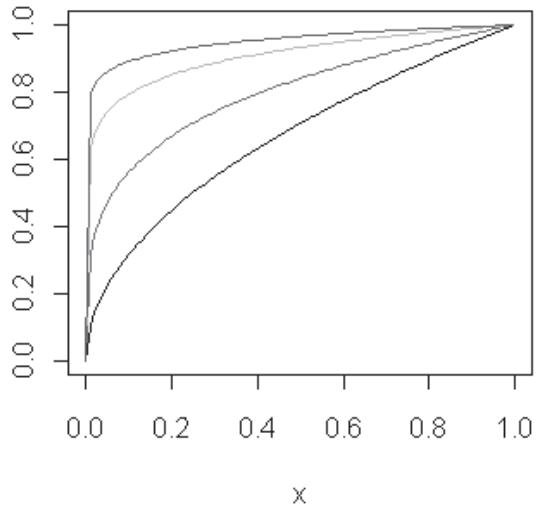

図 3.1　関数 x^b（b= 0.05（青）,b= 0.1（緑）,b= 0.25（赤）,b= 0.5（黒））のグラフ

この関数はパラメータ b が 0 に近づくほど急激に増加する特徴をもっている。一方，本論文で定義したシグモイド関数は図 3.2 からパラメータ b が 0 に近づくほどシグモイド関数は $[0,1]$ 上のほとんど全ての点で $\frac{1}{2}$ の値をとることが分かる。シグモイド関数を用いて SQL インジェクション攻撃の検出を行う場合，入力 x が攻撃文字列である確率を $\frac{x^b}{x^b+(1-x^b)}$ で表されるが，$0<b<1$ のどのようなパラメータ b を選んでも $\left[0,\frac{1}{2}\right]$ 上の点では $\frac{1}{2}$ 以下の値をとる，

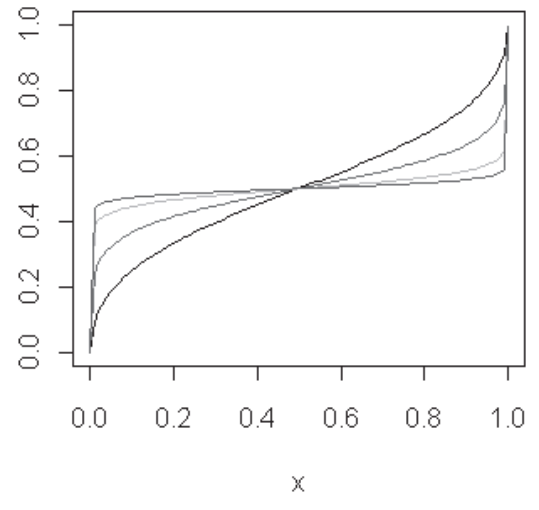

図 3.2　関数 $\frac{x^b}{x^b+(1-x^b)}$（b= 0.05(青) ,b= 0.1(緑) ,b= 0.25(赤) ,b= 0.5(黒)）のグラフ

つまり，入力文字列から計算されるxの値が$\frac{1}{2}$であるとき，入力された文字列が攻撃文字列である確率は$\frac{1}{2}$以下であることを表すことになる。しかしながら，文献[9][10]や2章の例2.3や例2.4で示したように，多くの攻撃文字列に対するxの値は$\frac{1}{2}$より小さな値をとることが多いため，シグモイド関数を利用したモデルで攻撃検出を行うと誤検出が起こる可能性が高くなると考えられる。また，攻撃文字列に対してxの値が小さな値をとるということは攻撃文字列に含まれる文字が正常文字列にあまり含まれないことを意味しており，2章の終わりに述べた通り，文献[10]で得られたxの値が0.08以上であれば攻撃文字列としてよいという結果と照らし合わせると，関数x^bはSQLインジェクション攻撃の特徴を捉えた関数の一種であると考えられる。これらのことから，我々の提案モデルに利用した関数x^bはシグモイド関数と比較して，xが小さな値をとる場合でも攻撃検出の確率を高くできるという意味でSQLインジェクション攻撃検出に有効であるといえる。

4. まとめ

本論文では，SQLインジェクション攻撃の自動検出を支援するための数理モデルを提案した。また，数理モデルの予測性能を示す予測誤差を定義し，2値判別の問題で広く応用されているシグモイド関数と比較することで，提案モデルの有効性を示した。

今後の課題として，関数x^bの最適なパラメータbを定めるための学習アルゴリズムの開発，SQLインジェクション攻撃を特徴づける他の関数の検討の他，提案モデルを他のWebアプリケーション攻撃に適用することなどが挙げられる。

参考文献

(1) NT Web Technology Vulnerabilities [online]：http://www.phrack.com/issues.html?issue= 54

(2) ソフトウェア等の脆弱性関連情報に関する届け出状況（2010 年第四半期），独立行政法人 情報処理推進機構［online］：http://www.ipa.go.jp/security/vuln/report/vuln 2010 q 4 .html

(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］，（45），pp. 103 - 108，（2008）

(4) William Robertson, Giovanni Vigna, Christopher Kruegel, Richard A. Kemmerer：“Using Generalization and Characterization Techniques in the Anomaly-based De-tection of Web Attacks”，In Proceeding of the Network and Distributed System Security（NDSS）Symposium, San Diego, CA, February,（2006）.

(5) Fredrik Valeur, Darren Mutz, Giovanni Vigna：“A Learning-Based Approach to the Detection of SQL Attacks”，［online］, http://www.cs.ucsb.edu/ ～ vigna/publications/ 2005 valeur mutz vigna dimva 05 .pdf,（2005）

(6) Engin Kirda, Christopher Kruegel, Giovanni Vigna, Hennaed Jovanovic,“Noxes: A Client-Side Solution for Mitigating Cross Site Scripting Attacks”, Security Track of the 21 st ACM Symposium on Applied Computing（SAC 2006）, Dijon, France, April,（2006）

(7) Davide Ariu, Giorgio Giacinto：“HMMPayl: an application of HMM to the analysis of the HTTP Payload”，JMLR: Workshop and Conference Proceedings 11 , pp. 81 – 87 ,（2010）

(8) Frank S. Rietta：“Application layer intrusion detection for SQL injection”，ACM-SE 44 Proceedings of the 44 th annual Southeast regional conference,（2006）

(9) Takeshi Matsuda, Daiki Koizumi, Michio Sonoda, and Shigeichi Hirasawa：“On Predictive Errors of SQL Injection Attack Detection by the Feature of the Single Character”，Proceeding of 2011 IEEE International Conference on Systems, Man, and Cybernetics（to appear）

(10) Michio Sonoda, Takeshi Matsuda, Daiki Koizumi and Shigeichi Hirasawa：“On Automatic Detection of SQL Injection Attacks by the Feature Extraction of the Single Character”，Proceeding of 2011 International Conference on Security of Information and Networks, ACM（to appear）

(11) I.S.Isa, Z.Saad, S.Omar, M.K.Osman,K.A.Ahmad and H.A.Mat Sakim：“Suitable MLP Network Activation Functions for Breast Cancer and Thy-roid disease Detection”，In proceedings of Second International Conference on Computational Intelligence, Modeling and Simulation（2010）

(12) T. Takiguchi, A. Sako, T. Yamagata and Y. Ariki：“System Request Utterance Detection Based on Acoustic and Linguistic Features”，I-Tech Education and Publishing, pp. 539 - 550 ,（2008）

(13) Jeong Hoon Ko, Jung Suk Joo, and Yong Hoon Lee：“On the Use of Sigmoid Functions for Multistage Detection in Asynchronous CDMA Systems”，IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, VOL. 48 , No. 2 , pp. 522 - 526 ,（1999）

(14) W. G. Halfond, Viegas and A. Orso：“A Classification of SQL Injection Attacks and Countermeasures”，College of Computing Georgia Institute of Technology IEEE,（2006）

(15) A. Tajpour, M. Masrom, M.Z. Heydari and S. Ibrahim：“SQL injection detection and prevention tools assessment”，2010 3 rd IEEE International Conference on Computer Science and Information Technology（ICCSIT 2010）, pp. 518 - 522 ,（2010）

Discussion on SQL Injection Attacks Detection Algorithm and Mathematical Model

Takeshi Matsuda

Abstract

SQL injection attacks are big problem to the security of database driven applications. The prevention methods against SQL injection attacks, such as parsing and pattern recognition, have been developed, but these techniques may not cope with the evasion of SQL injection attacks detection. Instead of these methods, Web application firewalls (WAF) are widely using to prevent Web application, generally. WAF detects SQL injection attacks using black list. However, a big black list may result in lowering of the detection performance. In this paper, we proposed a mathematical model to detect SQL injection attacks, and defined prediction error to measure the performance of mathematical models. Then we compared our proposed model with the mathematical model applying sigmoid function, and showed that the prediction error of our proposed model is less than the sigmoid model.

Keywords: SQL injection attacks, detection, mathematical model