Vol.2011-DPS-146 No.49
Vol.2011-CSEC-52 No.49
2011/3/11

文字単位の特徴抽出による SQL インジェクション攻撃検出法について

園田道夫[†1]　松田健[†1]
小泉大城[†1]　平澤茂一[†1]

従来の研究においては，SQL インジェクション攻撃の検出は構文解析や過去の攻撃手法をブラックリスト化したものが広く使われている．しかし，これらのアプローチでは，多様化する攻撃の偽装工作への対応が困難になってきている．そこで本研究では，入力文字列のサンプルとして攻撃と正常の二種類用意した．そして，攻撃を特徴づける文字が入力された文字列の中に占める割合をもとに両方の誤検出率を小さくし，かつ高速なアルゴリズムを提案し，評価を行った．

On the Automatic Detection of the SQL Injection Attacks by the Feature Extraction of the Single Character

Michio Sonoda,[†1] Takeshi Matsuda,[†1]
Daiki Koizumi[†1] and Shigeichi Hirasawa[†1]

In the conventional study, the parsing and the blacklist of the past attack methods are widely used in the detection of the SQL injection attack. However, the diversified camouflage technique makes detecting attacks difficult in such approaches. Then, in our study, we prepared the attack strings group and the normality strings group as samples of the input string, and we proposed high-speed algorithm, based on the ratio that occupied it to the character string where the character that characterize the attack was input, to reduce both misdetection rates, and we evaluated it.

†1 サイバー大学 IT 総合学部
Faculty of Information Technology and Business, Cyber University

1. はじめに

1.1 研究背景

Web アプリケーションへの攻撃は比較的新しく，99 年に電子雑誌 Phrack に rain forest puppy[1] によって発表された記事 *NT Web Technology Vulnerabilities*[2] で初めて網羅的に書かれ，ここで SQL インジェクション攻撃についても触れられている[⋆1]．SQL インジェクション攻撃とは，通常であれば Web アプリケーション内部で閉じているデータベースとのやりとりを，外部から直接行う攻撃である．リレーショナルデータベースを利用する場合，問い合わせ言語 SQL を用いてデータの検索や追加・削除などを行うが、外部の利用者の入力を適切に処理できていないと Web のフォームなどから SQL 文を入力してデータベースを操作できてしまう．

SQL インジェクション攻撃は，2005 年ごろから多数観測され始め，2006 年になると攻撃は急増し[3]，2007 年には自動拡散するワームやボットネットなどを利用した大規模攻撃へと移行していき[4]，現在でも継続的に多数の被害が出ている．プリペアドステートメントなどを用いてプログラムを開発するなどの根本的な SQL インジェクション対策は，技術的難易度は高くないものの普及はあまり進んでいない[5]．

Web アプリケーションの開発は，短納期かつ低予算で行われることが多いため，改修予算を確保しにくく，業界の性質上人の入れ替わりも激しく行われるため，知見が蓄積されにくい現状がある．暫定的な対策としては，Web アプリケーションファイアウォール（WAF）があるが，商用のものは高価であり導入しにくく，非商用のものは限定的にしか利用されていない．また，新たな攻撃方法の研究が盛んに行われているため，攻撃そのものをブラックリスト化して検知しようとする WAF では持続的な対応が難しくなってきている．

1.2 従来研究

これまで，Web アプリケーションそのものを修正せずに保護する場合には，WAF が利用されてきた．WAF は基本的にはブラックリスト方式であり，新たな攻撃手法が開発されるとブラックリストのデータベースを更新しなければならない仕組みである．WAF はフリーウェアとしても存在する[6][7] が，新たな攻撃手法に追随するためにデータベースを手作業でメンテナンスしなければならないので，持続的な対応が難しい．

従来の実装が主に過去の攻撃を集積したブラックリストに依る検出であったために，研究

⋆1 攻撃の名称として SQL インジェクションという言葉は用いられてはいないが，ほぼ同義である．

Vol.2011-DPS-146 No.49
Vol.2011-CSEC-52 No.49
2011/3/11

はそれ以外のアプローチを模索しているものが多く，SQL インジェクション攻撃に限れば，Web アプリケーションに渡されるパラメータのアルファベット数や数字数，文字数などを特徴量とした異常検知手法の提案もなされている[8]．

異常検知手法は他にも研究が行われているが，異常を判定する尺度となる正常値を学習する必要があるもの[9][10][11][12] が多く，例えば Web サイトを頻繁にリニューアルする場合などにおいては運用上の課題が残る．また，攻撃に用いられる文字列の構文解析を行った後解析するモデルも存在する[13]．検出精度を上げるために構文解析的手法を採用する研究も多いが，構文解析自体が性能上の課題となる．

1.3 研究目的

本研究では，攻撃手法ブラックリストに依存せず，かつ構文解析を行わずに Web アプリケーションへの攻撃，特に SQL インジェクション攻撃を機械的に検出することを目的とする．特に攻撃文字列に含まれる記号文字に着目し，その出現頻度によって攻撃かどうかを見極めるが，その点ではパラメータに含まれるアルファベット数や数字数，文字数などに着目した異常検知手法[8] に近いアプローチである．同研究が本研究と異なるのは，パラメータに含まれる文字すべてを計算対象としているところになる．

以下，本論文の構成について述べる．次の第 2 章では，SQL インジェクション攻撃を文字単位の特徴抽出によって検出する方法について概要を述べ，提案アルゴリズムについて説明する．続く第 3 章では，提案アルゴリズムの攻撃検出の性能を，人工的に発生させた正常および攻撃文字列を処理させた場合の検出率によって評価し，考察を行う．最後の第 4 章で結論と今後の課題について述べる．

2. 文字単位の特徴抽出による SQL インジェクション攻撃の検出

SQL インジェクション攻撃は Web ページ上のフォームに攻撃文字列を入力して行うものである．この章では, 本研究で提案する, 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ためのアルゴリズムを紹介する．

2.1 準　　備

フォームに入力された入力文字列を l とおき, 入力文字列全体からなる集合を L で表す．また, 入力文字列 l の文字列長を $|l|$ と定義する．本研究では, フォームに入力された l が攻撃文字列であるか正常文字列であるかを l に含まれる文字に基づいて行うため，l に含まれるすべての記号や文字のうち, 攻撃を特徴づけるための文字を s で表す．

s の候補としては，表 1 に示す計 20 種類の文字を考慮した．

表 1　予備実験に用いた攻撃を特徴づける文字の候補
Table 1　Candidates of Characters for Feature Extraction of Attacks

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}	アンパサンド (&)
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_{xiv}	アスタリスク (∗)
s_{x}	シャープ (#)	s_{xx}	スラッシュ (/)

2.2 攻撃を特徴づける文字集合の決定

本節では, 前節で述べた 20 文字の候補 $s_{\mathrm{i}}, s_{\mathrm{ii}}, \cdots, s_{\mathrm{xx}}$ の中で, SQL インジェクション攻撃の検出に有効な文字を選択することを目的に行った予備実験について説明する．

いま，20 個の候補の文字 $s_{\mathrm{i}}, s_{\mathrm{ii}}, \cdots, s_{\mathrm{xx}}$ について, 各文字が入力文字列 l に含まれる総数をそれぞれ $\#s_{\mathrm{i}}, \#s_{\mathrm{ii}}, \cdots, \#s_{\mathrm{xx}}$ で定義する．

また, 0 から 1 までの有理数の集合を $[0,1]_{\mathbf{Q}} \subset \mathbf{Q}$ と定義する．このとき, 集合 L から $[0,1]_{\mathbf{Q}}$ への写像 $f : L \to [0,1]_{\mathbf{Q}}$ を,

$$f(l) = \frac{\#s_a}{|l|}, \tag{1}$$

で定義する．この写像は $[0,1]_{\mathbf{Q}}$ の要素が与えられると, それに対応する入力文字列 l が少なくとも 1 つ構成できるが, $l_i \neq l_j$ でも $f(l_i) = f(l_j)$ となる場合があるため, $\#s_a/|l|$ から入力された文字列を特定することは出来ない．したがって, $\#s_a/|l|$ から入力された文字列を特定することはできない．

本論文では, 入力文字列 l_i に対して, $\#s_a/|l| \in [0,1]_{\mathbf{Q}}$ を攻撃を特徴づける文字の占有率とよび,

$$p_{ia} = \frac{\#s_a}{l_i}, \tag{2}$$

と定義する．

Vol.2011-DPS-146 No.49
Vol.2011-CSEC-52 No.49
2011/3/11

表 2 攻撃文字列のサンプル
Table 2 Samples of SQL Injection Attack Sequences

番号	攻撃文字列
1	`DROP sampletable;--`
2	`';DROP sampletable;--`
3	`;DROP sampletable;--`
4	`DROP sampletable;#`
⋮	⋮
623	`test'||'`
624	`'||'`

表 3 正常文字列のサンプル
Table 3 Samples of SQL Normal Sequences

番号	正常文字列
1	test
2	password
3	`sonod@`
4	`@sonoda`
⋮	⋮
231	{(1%2) + (3/4)}/5
232	& nicovideo(表示させたいニコニコ動画の URL) { width,height }

2.2.1 攻撃文字列および正常文字列のサンプルの設定

文献[13)14)15)16)17)] から収集した攻撃文字列集合を生成した (624 個)．この集合の各要素は SQL 文の中で特別な意味を持つ区切り記号など，単独で攻撃を励起すると思われる記号や，攻撃文字列で頻出すると想定される記号，SQL 文のコマンドなどで構成される文字列である．

さらに，一般的な Web の入力フォームを想定し，攻撃ではない入力文字列のグループを作成した（234 個）．このグループは ID，パスワードや住所氏名などを想定して生成した，全角，半角文字や記号などを混ぜたサンプル，ブログなどで多用される Wiki 文法に基づいたサンプル，およびネットで多用される顔文字のサンプルなどで構成されている．

攻撃文字列の集合を D_A, 正常文字列の集合を D_N で表し, $D = D_A \cup D_N$ と定義する．表 2 および表 3 に，攻撃文字列および正常文字列のサンプルの一部を示す．

2.2.2 予 備 実 験

集合 D に含まれるすべての l_i に対し，占有率 p_{ia} を計算し, $0 \leq \alpha \leq 1$ を満たす実数 α に対して, $p_{ia} > \alpha$ を満足する $l_i \in D_A$ の $|D_A|$ に対する割合,

$$P(D_A, a) = \frac{\#\{i | l_i \in D_A, p_{ia} \neq 0\}}{|D_A|}, \tag{3}$$

と, $p_{ia} \leq \alpha$ を満足する $l_i \in D_N$ の $|D_N|$ に対する割合,

$$P(D_N, a) = 1 - \frac{\#\{i | l_i \in D_N, p_{ia} \neq 0\}}{|D_N|}, \tag{4}$$

を計算し，攻撃文字列を特徴づける文字の選択に利用した．

表 4 $s_a, a = \mathrm{i}, \mathrm{ii}, \cdots, \mathrm{xx}$ についての $P(D_A, a), P(D_N, a)$ の値
Table 4 Result of Preliminary Simulation

文字候補	$P(D_A, a)$ の値	$P(D_N, a)$ の値
s_{i} 半角スペース	0.971	0.098
s_{ii} セミコロン	0.664	0.000
s_{iii} シングルクォーテーション	0.483	0.000
s_{iv} 右側丸括弧	0.459	0.060
s_{v} 左側丸括弧	0.414	0.090
s_{vi} 右側中括弧	0.000	0.060
s_{vii} 左側中括弧	0.000	0.026
s_{viii} 右側大括弧	0.000	0.026
s_{ix} 左側大括弧	0.000	0.026
s_{x} シャープ	0.032	0.021
s_{xi} パーセント	0.024	0.026
s_{xii} ダブルクォーテーション	0.010	0.000
s_{xiii} アンパサンド	0.019	0.021
s_{xiv} バックスラッシュ	0.032	0.000
s_{xv} パイプ	0.040	0.103
s_{xvi} 等号	0.294	0.060
s_{xvii} 大なり不等号	0.000	0.090
s_{xviii} 小なり不等号	0.000	0.038
s_{xix} アスタリスク	0.128	0.094
s_{xx} スラッシュ	0.112	0.073

2.2.3 予備実験の結果

20 種類の文字候補 $s_{\mathrm{i}}, s_{\mathrm{ii}}, \cdots, s_{\mathrm{xx}}$ のそれぞれについて，表 2 の攻撃文字列のサンプル，および表 3 の正常文字列のサンプルに対する式 (3), (4) の値を計算した結果を表 4 に示す．

2.2.4 予備実験の考察

攻撃を特徴づける文字 $s_a (a = \mathrm{i}, \mathrm{ii}, \cdots, \mathrm{xii})$ のうち, $P(D_A, a)$ の値が大きく, かつ $P(D_N, a)$

Vol.2011-DPS-146 No.49
Vol.2011-CSEC-52 No.49
2011/3/11

の値が小さな文字が, 攻撃検出に適した文字といえる.

表 4 より，攻撃文字列中に占める割合の中では $P(D_A, \mathrm{i})$ が最も値が大きく, 攻撃検出に最も有効な文字であると考えられる. しかしながら, 同時に正常文字列中に占める割合である $P(D_N, \mathrm{i})$ の値も $P(D_N, a)$ の中で最大であるため, s_{i} は正常文字列を攻撃文字列であると誤検出する可能性が高い.

一方, $s_{\mathrm{ii}}, s_{\mathrm{iii}}$ については $P(D_A, \mathrm{ii}), P(D_A, \mathrm{iii})$ の値がともに $P(D_A, \mathrm{i})$ に次いで高く, かつ $P(D_N, \mathrm{ii}), P(D_N, \mathrm{iii})$ の値がともに最小となっているため, これらの文字を用いて攻撃検出を行えば, 正常文字列を攻撃文字列であると誤検出する可能性が低くなると考えられる. そのため, 攻撃検出には $P(D_A, a)$ の値が最大である s_{i} のみを用いるのでなく, s_{i} といくつかの他の $P(D_A, a)$ が大きく, かつ $P(D_N, a)$ が小さな文字を組み合わせた文字集合で攻撃検出するを行うことで, 攻撃の検出率を上げ, かつ誤検出率をさげることが可能になると考えられる.

SQL インジェクション攻撃は文字通り直接 SQL 文をフォームに入力して行う攻撃であるため, 攻撃文字列中には SQL 文の区切り文字が頻出する. 区切り文字として最も多用されるのものは半角スペース s_{i} である. また, 文字 $s_{\mathrm{ii}}, s_{\mathrm{iii}}$ は文字列をプログラム内での文字型変数に定数として格納する際に用いられる記号であり, Web の入力フォームから入力されてくる文字列の処理が適切でない場合, 入力文字列の強制的な終端として意味を持ってしまう. したがって, SQL インジェクション攻撃の攻撃文字列の最初の部分もしくはそれに近いところに 1 個は出現する.

以上の考察を踏まえ，攻撃の特徴を抽出できると思われる文字を選択したところ，以下のようになった.

- s_{i} 半角スペース
- s_{ii} セミコロン
- s_{iii} シングルクォーテーション
- s_{iv} 右側丸括弧
- s_{v} 左側丸括弧

2.3 提案アルゴリズム

予備実験の結果より，5 つの文字 $s_{\mathrm{i}}, s_{\mathrm{ii}}, s_{\mathrm{iii}}, s_{\mathrm{iv}, s_{\mathrm{v}}}$ からなる攻撃を特徴づける文字集合を,

$S_1 = \{s_{\mathrm{i}}, s_{\mathrm{ii}}, s_{\mathrm{iii}}\}, S_2 = \{s_{\mathrm{i}}, s_{\mathrm{ii}}, s_{\mathrm{iv}}\},$
$S_3 = \{s_{\mathrm{i}}, s_{\mathrm{ii}}, s_{\mathrm{v}}\}, S_4 = \{s_{\mathrm{i}}, s_{\mathrm{iii}}, s_{\mathrm{iv}}\},$
$S_5 = \{s_{\mathrm{i}}, s_{\mathrm{iii}}, s_{\mathrm{v}}\}, S_6 = \{s_{\mathrm{i}}, s_{\mathrm{iv}}, s_{\mathrm{v}}\},$
$S_7 = \{s_{\mathrm{ii}}, s_{\mathrm{iii}}, s_{\mathrm{iv}}\}, S_8 = \{s_{\mathrm{ii}}, s_{\mathrm{iii}}, s_{\mathrm{v}}\},$
$S_9 = \{s_{\mathrm{ii}}, s_{\mathrm{iv}}, s_{\mathrm{v}}\}, S_{10} = \{s_{\mathrm{iii}}, s_{\mathrm{iv}}, s_{\mathrm{v}}\},$
$S_{11} = \{s_{\mathrm{i}}, s_{\mathrm{ii}}, s_{\mathrm{iii}}, s_{\mathrm{iv}}\}, S_{12} = \{s_{\mathrm{i}}, s_{\mathrm{ii}}, s_{\mathrm{iii}}, s_{\mathrm{v}}\},$
$S_{13} = \{s_{\mathrm{i}}, s_{\mathrm{ii}}, s_{\mathrm{iv}}, s_{\mathrm{v}}\}, S_{14} = \{s_{\mathrm{i}}, s_{\mathrm{iii}}, s_{\mathrm{iv}}, s_{\mathrm{v}}\},$
$S_{15} = \{s_{\mathrm{ii}}, s_{\mathrm{iii}}, s_{\mathrm{iv}}, s_{\mathrm{v}}\}, S_{16} = \{s_{\mathrm{i}}, s_{\mathrm{ii}}, s_{\mathrm{iii}}, s_{\mathrm{iv}}, s_{\mathrm{v}}\}$

と定義し, 集合 S_k に属する文字が入力文字列 l に出現する総数を $\#S_k$ で定義する. ただし, $k = 1, 2, \cdots, 16$ である. このときの攻撃を特徴づける文字の占有率を,

$$p_{ik} = \frac{\#S_k}{l_i} \in [0, 1]_{\mathbf{Q}}\,, \tag{5}$$

と定義する.

本論文では, 入力文字列 l が攻撃文字列であるか正常文字列であるかを判別するための関数 $h : [0, 1] \to \{0, 1\}$ を

$$h(p_{ik}) = \begin{cases} 1 & (p_{ik} > \alpha); \\ 0 & (p_{ik} \leq \alpha)\,, \end{cases} \tag{6}$$

と定義し, $h(p_{ik}) = 1$ のとき l_i を攻撃文字列, $h(p_{ik}) = 0$ のとき l_i を正常文字列と判別する.

ただし, $[0, 1]$ は 1 次元ユークリッド空間 $\mathbf{R}$ の閉区間を表し, 定数 α は判別を行うための閾値を表すものとする.

本論文では, サンプル D から n 個の攻撃文字列 $\{l_1, l_2, \cdots, l_n\}$ と m 個の正常文字列 $\{l_{n+1}, l_{n+2}, \cdots, l_{n+m}\}$, 合計 $n + m$ 個の文字列をランダム抽出し, 以下の手順で攻撃検出を行うための文字集合 S_k $(k = 1, 2, \cdots, 16)$ を選択し, 閾値 α を決定する. そのために以下の 2 つの記号

$$x_{kj} = \frac{\sum_{i=1}^{n} p_{ik}}{n}\,, \tag{7}$$

$$y_{kj} = 1 - \frac{\sum_{i=1}^{n} p_{ik}}{n}\,, \tag{8}$$

Vol.2011-DPS-146 No.49
Vol.2011-CSEC-52 No.49
2011/3/11

を定義する.

x_{kj} は，攻撃文字列 $\{l_1, l_2, \cdots, l_n\}$ から定まる占有率 p_{ik} の中で $p_{ik} > \alpha$ を満足するものの割合を示すものであり, x_{kj} の値が 1 に近いほど検出率が高くなることが分かる.

同様に, y_{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が分かる. l_i $(i = 1, 2, \cdots, n+m)$ については攻撃文字列であるか, 正常文字列であるかについては既知であることに注意する.

提案アルゴリズム

(1) J 個の閾値候補 $0 \leq \alpha_j \leq 1\,(j = 1, 2, \cdots, J)$ を固定する

(2) すべての i, k に対して, x_{kj}, y_{kj} を計算し, これを R 回繰り返す α_j を選び, 特徴抽出を行う文字集合 S_k を決定する.

(3) すべての S_k に対して, $\sum_{r=1}^{R} x_{kjr}/R \geq 0.9$ を満足する α_j を選んでできる集合を A_k とする.

(4) A_k のうち, $\max_j\{(\sum_{r=1}^{R} x_{kjr}/R) + \beta(\sum_{r=1}^{R} y_{kjr}/R)\}$ となる閾値 α_j を選び, 特徴抽出を行う文字集合 S_k を決定する.

本論文では, $\sum_{r=1}^{R} x_{kjr}/R$ のことを攻撃検出力, 定数 $\sum_{r=1}^{R} y_{kjr}/R$ を正常検出力と呼ぶ.

3. 提案法の攻撃検出率の実験による評価

3.1 実　　験

予備実験で抽出した $S_1, S_2, \cdots, S_{16}$ の文字集合について，攻撃文字列と正常文字列をランダムに生成させ，攻撃検出力と正常検出力を計算した．このとき，攻撃文字列は 20〜200 個まで，正常文字列は 20〜100 個までのさまざまな一様分布で生成させ，閾値 α_j は 0.01 から 0.15 まで変化させた.

3.1.1 実　験　1

$S_1, S_2, \cdots, S_{16}$ について，攻撃検出力と正常検出力の加重平均の重み (提案アルゴリズム (4) の β) を定め，様々な閾値 α_j に対して，両者の加重平均の値を計算した.

3.1.2 実　験　2

$S_1, S_2, \cdots, S_{16}$ について，閾値 α を 0.01 から 0.15 まで変化させ，攻撃検出力に対する正常検出力の重みを 0 から 10 まで増やしながら，両者の加重平均の値を計算した.

3.2 実 験 結 果

実験 1 の結果の一例として，文字集合 S_1 について，攻撃検出力と正常検出力の加重比 1:5 とした場合に，横軸に閾値，縦軸に攻撃検出力と正常検出力の加重平均をプロットした図を図 1 に示す．ただし，ここでは，正常文字列を 40 個，攻撃文字列を 150 個サンプリングし，これを 200 回繰り返した算術平均を取ることで正常検出力と攻撃検出力を計算している.

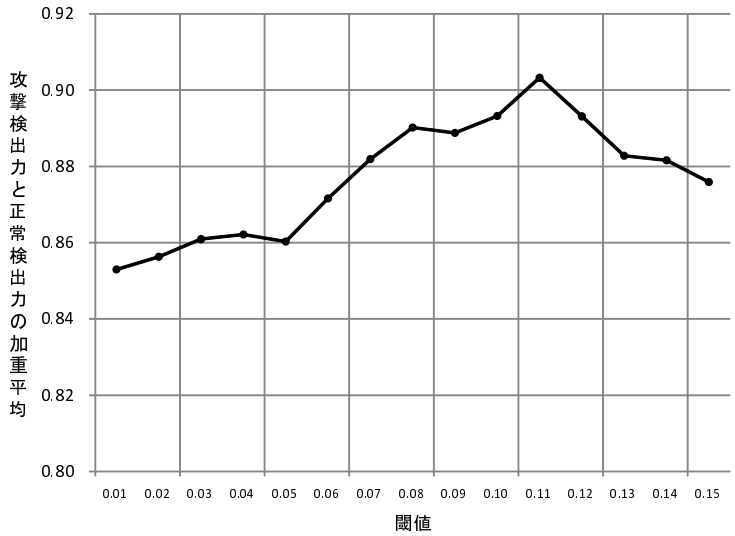

図 1　攻撃検出力と正常検出力の加重平均と閾値の関係 (加重比 1 : 5，文字集合：S_1)

また，実験 2 の結果の一例として，文字集合 S_2 について，閾値 0.01 と 0.08 の場合に，横軸に攻撃文字列に対する正常文字列の重みを，縦軸に攻撃検出力と正常検出力の加重平均をプロットした図を図 2 に示す．なお，攻撃検出力と正常検出力の加重比は 1:5 としており，ここでも，正常文字列を 40 個，攻撃文字列を 150 個サンプリングし，これを 200 回繰り返した算術平均を取ることで正常検出力と攻撃検出力を計算している.

3.3 考　　察

図 1 では，閾値 0.11 を頂点としてほぼ上に凸のグラフが得られた．攻撃検出力のみを対象として，閾値を変化させた場合，グラフは文字集合を問わず単調減少となった．すなわち，攻撃検出だけに特化できるのであれば，最適な閾値は 0.01 となる．しかし，攻撃検出と正常検出の両方を加味し，その加重比が 1:5 とした図 1 では，閾値が 0.11 のときがもっとも（加重）検出力が高くなっている．このように，正常文字列と攻撃文字列の生成する分布によっては，閾値を適切に設定することで，攻撃文字列の検出のみを対象とする手法より

Vol.2011-DPS-146 No.49
Vol.2011-CSEC-52 No.49
2011/3/11

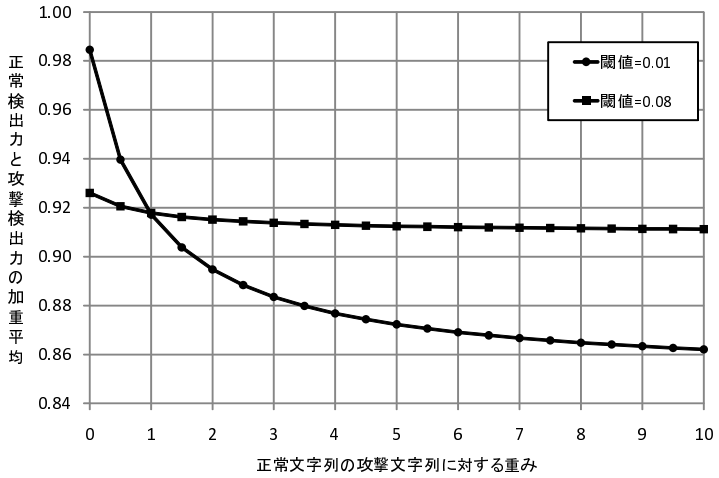

図 2 攻撃検出力と正常検出力の加重平均と正常文字列の攻撃文字列に対する重みとの関係 (文字集合：S_2)

も提案法のほうが検出力が高い場合があることを確認することができた．

一方，図 2 では，横軸に正常文字列の攻撃文字列に対する重みを，縦軸に攻撃検出力と正常検出力の加重平均（加重比 1:5）をプロットした．図 2 では，閾値 0.01 の場合は，正常文字列の重みが増えると検出力の加重平均が 0.86 付近まで下がっていくのに対し，閾値 0.08 の場合は，正常文字列の重みを問わず，0.90 以上でほぼ変わらない検出力となっているのがわかる．攻撃文字列のみが入力されるとわかっているのであれば，閾値 0.01 を選ぶことで攻撃検出力が 0.98 程度となるような高い攻撃検出を行うことができるが，実際には正常文字列も存在しているため，これは現実的ではない．提案法では，攻撃文字列と正常文字列の両方を考慮して適切な閾値を設定することで，たとえ正常文字列と攻撃文字列の生成する分布が様々に変わっても，安定した検出が実現可能であることが見て取れる．

SQL インジェクション攻撃の場合，攻撃は主に Web ページの入力フォームへの入力を用いて行われるが，提案法ではそのフォームへの入力文字列に対するこの文字グループの占有率を都度調査し，占有率が閾値を上回れば攻撃として検出させる．今後実際にソフトウエア開発を行い実運用で検証していく計画だが，攻撃手法ブラックリストとの照合や構文解析に比べて運用上の計算量は少なく，検出や判定は高速に行えると想定される．

4. 結論および今後の課題

本論文では，文字単位の特徴抽出による SQL インジェクション攻撃の検出アルゴリズムを提案し，人工データを用いた評価を行った．提案法では，攻撃文字列および正常文字列のサンプルデータを用いて，攻撃検出と正常検出の両方を加味した文字集合および既知の閾値を用いるが，入力文字列中に文字集合の要素が含まれる頻度を数えるという，単純なアルゴリズムであるため，高速に攻撃検出を行うことができる．

提案法の課題は，攻撃文字列と正常文字列の収集・生成である．今回実験材料として文献や Web 上の情報などを利用したが，本来ならば実際に運用されている Web サイトで攻撃を観測して収集すべきであろう．しかし，実際には機械的な攻撃が多いため実サイト運用で収集できる攻撃文字列のデータは種類が少なく，攻撃となり得るパターンを網羅しきれない．そのため今回は，なかなか観測されなくてもあり得るデータを含めて人工的なデータを収集，生成したが，実サイトでの収集を含むサンプルの取得方法を考える必要がある．効率性を考慮するなら，収集は自動化していく必要があると思われる．

また，今回は攻撃を特徴づける文字群が正常文字列中に出現する頻度が低いモデルとして SQL インジェクション攻撃のみを対象としたが，正常文字列中に出現する頻度が比較的高くなるクロスサイトスクリプティング攻撃も対象として，同じ手法による検出の有効性を調査，検証していく予定である．

参 考 文 献

1) rain forest puppy [online], http://www.wiretrip.net/rfp/
2) NT Web Technology Vulnerabilities [online], http://www.phrack.com/issues.html?issue=54
3) 侵入傾向分析レポート Vol.7, JSOC Analysis Team [online], http://www.lac.co.jp/info/jsoc_report/pdf/20061030lac_report.pdf, 2006 年 10 月.
4) 侵入傾向分析レポート Vol.10, JSOC Analysis Team [online], http://www.lac.co.jp/info/jsoc_report/pdf/20080415lac_report.pdf, 2008 年 4 月.
5) 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/vuln2010q4.html, 2011 年 1 月.
6) ModSecurity, Breach Security, Inc. [online], http://www.modsecurity.org/
7) Guardian@JUMPERZ.NET [online], http://guardian.jumperz.net/index.html

Vol.2011-DPS-146 No.49
Vol.2011-CSEC-52 No.49
2011/3/11

8) 角田直樹，安井浩之，松山実，異常検出手法を用いた SQL インジェクション攻撃の検出，全国大会講演論文集 第 71 回平成 21 年 (3), ”3-379”-”3-380”, 2009-03-10.
9) William Robertson, Giovanni Vigna, Christopher Kruegel, Richard A. Kemmerer, “Using Generalization and Characterization Techniques in the Anomaly-based Detection of Web Attacks,” in Proceeding of the Network and Distributed System Security (NDSS) Symposium, San Diego, CA, February, 2006.
10) 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_dimva05.pdf, 2005.
11) Engin Kirda, Christopher Kruegel, Giovanni Vigna, Hennaed Jovanovic, “Noxes: A Client-Side Solution for Mitigating Cross Site Scripting Attacks”, Security Track of the 21st ACM Symposium on Applied Computing (SAC 2006), Dijon, France, April, 2006.
12) Davide Ariu,Giorgio Giacinto, “HMMPayl: an application of HMM to the analysis of the HTTP Payload”, JMLR: Workshop and Conference Proceedings 11 (2010) pp.81–87, 2010.
13) Frank S. Rietta, “Application layer intrusion detection for SQL injection”, ACM-SE 44 Proceedings of the 44th annual Southeast regional conference, 2006.
14) Justin Clarke, *SQL Injection Attacks And Defense,* Syngress Publishing Inc., 2009.
15) “SQL Injection Cheat Sheet” [online], http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
16) “SQL Injection Cheat Sheet” [online], http://michaeldaw.org/sql-injection-cheat-sheet
17) “MySQL SQL Injection Cheat Sheet” [online], http://pentestmonkey.net/blog/mysql-sql-injection-cheat-sheet