Vol.2015-CSEC-71 No.16
2015/12/4

文字の出現頻度とそれらの関連性による SQL インジェクション攻撃検出法

佐野綾子[†1]　園田道夫[†2]　松田健[†3]　趙晋輝[†4]

概要：SQL インジェクション攻撃はサイバー攻撃の一種であり，この攻撃を行うことで攻撃者はデータベースから様々な個人情報を引き出し，悪用することが出来る．現在この攻撃に対する対策はいくつもあるが，普及までに時間がかかる，もしくは新たな攻撃手法に対応できないなどの問題を抱えている．そこで，本研究では未知の攻撃も含め攻撃文のみを高い精度で検知することを目的に，N-gram 解析，Skip-gram 解析を用いたアルゴリズムを提案する．

キーワード：SQL インジェクション攻撃，N-gram 解析，Skip-gram 解析，ベイズの定理

SQL injection attack detection by frequency analysis

AYAKO SANO[†1]　MICHIO SONODA[†2]
TAKESHI MATSUDA[†3]　JINHUI CHAO[†4]

Abstract: SQL Injection attack is a kind of cyber attacks in which attackers can steal private data from Data Base in Web applocations. Several detection algorithms of attack have been proposed. However, it seemed that it is hard to deal with unknown attacks with new variations. In this paper, we propose a new algorithm using N-gram analysis and Skip-gram analysis in order to detect unknown attack with high precision.

Keywords: SQL Injection Attack，N-gram analysis，Skip-gram analysis，Bayes' theorem

1. はじめに

1.1 研究背景

近年，インターネットを通じて受けられるサービスが急速に多様化し，増加している．具体的には Twitter, Facebook などの各 SNS や amazon，楽天に代表されるようなインターネットショッピング，ネット上で金銭取引ができるインターネットバンキングなどが挙げられる．これらの Web サービスは，個人情報を登録し，アカウントを作成するだけで誰でも簡単に利用できる．また，通常であれば店舗が閉まっている時間帯であっても，インターネット環境さえあればどこにいても利用できるため利便性が大変高く，Web サービス利用者の数は年々急速に増加している．しかし，便利である一方，前述の通り，アカウント作成のためには個人情報を登録する必要があり，またサービス内容によっては金銭的なやり取りも発生するため，サイト管理者による個人情報漏えいのリスクも重くなっているといえる．そのため，近年では Web サービス提供側のセキュリティが重大視されている．

一口にセキュリティと言っても，情報漏えいの原因は主に二つにわけられる．社内の従業員や外注業者などによって情報が持ち出される内部要因と，悪意を持った第三者が Web サイトへ不正アクセスを行う外部要因である．

本研究で扱う SQL インジェクション攻撃は，外部要因の一つであり，データベースを利用している Web サイトに不正アクセスし，登録されている個人情報を持ち出す，破壊する，改ざんすることが可能な攻撃法である．これは Web サイト上で入力された文字列が，本来であれば単なるデータとして処理されるべきところをデータベースに問い合わせる際に使用される SQL 文の一部として認識されてしまうというバグを利用した攻撃である．

SQL インジェクション攻撃はかなり以前より観測されてきたサイバー攻撃であるが，2004 年ごろから攻撃者にとって使いやすいツールが普及したことから SQL インジェクションを利用した攻撃は急速に増加した．Web サービスを行う企業が SQL インジェクション攻撃にあい，クレジットカード情報などを含む大量の顧客情報が流出する事件が毎年いくつも起きている．いずれの事案に関しても，利用者にとっては自身の個人情報が流出し，悪用される危険性があり，企業側も顧客へお詫びや調査，ビジネスプロセスの見直し，セキュリティ強化対策など巨額の出費を余儀なくされた上，企業のブランドイメージにも重大な傷がつく事態を招き企業としての信頼を大きく損なっている．

†1 中央大学
Chuo University
†2 サイバー大学
Cyber University
†3 静岡理工科大学
Shizuoka Institute of Science and Technology
†4 中央大学
Chuo University

ⓒ2015 Information Processing Society of Japan

1.2 SQL インジェクション攻撃の例

SQL インジェクション攻撃の分類は例えば，文献[1]で行われているが，ここではその中からもっとも単純な SQL インジェクション攻撃の例を紹介する．なお，以下において表 1 のような SQL データベース内のテーブルがあると仮定する．

ユーザ番号	ユーザ名	パスワード
1	Aka	Red
2	Midori	Green
3	Kiiro	Yellow
4	Ao	Blue

表 1　テーブル名:　ユーザとパスワード

まずアプリケーション側で以下のような SQL 文を用意していたとする．

SELECT * FROM ユーザとパスワード WHERE ユーザ ID = ‘{$user}’ AND パスワード = ‘{$password}’

この Web サイトでは，ユーザによって入力されたユーザ ID，パスワードをそれぞれ{$user}，{$password}に代入し，SQL データベース内でそのような組み合わせが存在すればログイン出来る仕組みである．

さて，この Web サイトに対して悪意あるユーザが攻撃をする場合を考える．攻撃者がユーザ番号 2「Midori」として正しいパスワードを入力せずにログインしたい場合には以下のようなパラメータを入力する．

{$user}　Midori
{$password}　‘　OR　‘A’　=　‘A

結果として生成される SQL 文は次のようになる．
SELECT * FROM ユーザとパスワード WHERE ユーザ ID = ‘Midori’ AND パスワード = ‘‘ OR ‘A’ = ‘A’

入力値に「’」を含ませることでパスワードに関する WHERE 文の条件式をいったん終了させ，次の文に「‘A’ = ‘A’」という恒真式が対象となる OR 文を挿入している．結果 WHERE 文が常に真となり，正しいパスワードを入力することなくユーザ ID「Midori」としてログインすることが可能になる．なお，OR 文以降が常に真になるような恒真式であれば「‘A’ = ‘A’」の部分は何でもよい．

1.3 SQL インジェクション攻撃に対する対策

現在行われている対策としては，潜在的に悪意のある文字列を定めてエスケープ処理を行うブラックリスト方式や，安全であると考えられる文字のみを登録して使用を許可するホワイトリスト方式，特定の SQL 文テンプレートを用いることで不正な入力での SQL 文の改変をできないようにするプリペアドステートメントなどが挙げられる．

ブラックリスト方式では新しい攻撃が考え出された際に対応し終わるまでのギャップが生じてしまうことがある．また攻撃者によって以下のような対策を取られた場合期待された効果を発揮することが難しいという欠点がある．

- シングルクォートで囲まれていないフィールドを狙う．
- エスケープ処理される文字を使用せずに攻撃する．
- 本来ならばエスケープ処理されるべき文字を挿入し，それを隠蔽するためにストアドプロシージャを利用する．

ホワイトリスト方式，プリペアドステートメントはかなりの精度で SQL インジェクション攻撃を防ぐことが可能であり，コーディングの技術的な難易度も高くないものの現状として普及が進んでいない問題がある．

その要因として，Web アプリケーション開発を実際に行う上での制約が挙げられる．Web アプリケーションの開発はできるだけ短い納期で，かつ，可能な限り低予算で行われることが多いため，ホワイトリスト方式やプリペアドステートメントをコード内に組み込む，もしくは現在あるアプリケーションを改修する予算を確保しにくい．また，担当する人の入れ替わりが激しい業界であることからも，サイバー攻撃とその対策に関する知見が蓄積されにくいという問題がある．

根本的な対策が人的制約などによって行えない場合の暫定的な対策としては，主にブラックリスト方式を用いて攻撃の検知を行う Web アプリケーションファイアーウォール（WAF）である．しかし商業用の WAF はその多くが未だに高価であり，非商用の物やフリーソフトウェアとして頒布されている WAF は，新しい攻撃パターンが開発されるたびに手作業で更新する必要が出てきてしまう．前述のように，ブラックリスト方式は回避されてしまう可能性もあるため、現状の同方式による WAF のみでは SQL インジェクション攻撃に対する確実な対策ではないと考えられる．

2. 既存研究

2.1 その 1「文字集合からの特徴抽出による SQL インジェクション攻撃の自動検出における閾値学習アルゴリズム.」[3]

前述の SQL インジェクション攻撃対策の課題に対して，小泉ら[3]の研究では，攻撃文字列の記号ごとの頻度解析を行い，多く含まれるいくつかの半角記号に着目することで，これらの，文字列を攻撃文だと特徴づける記号が入力文字列中にどの程度含まれているかによって，ブラックリストに依存せず，新しい攻撃手法であっても高精度で検知できるアルゴリズムを提案している．

この手法では，入力された文字列を l として入力文字列全体からなる集合を L で表し，入力文字列の文字列の長さを $|l|$ と定義している．入力された文字列が攻撃文字列であるか通常文字列であるかは l に含まれる文字から判別するが，その際に使用する文字列を攻撃文字列であると特徴づける記号として，以下に示す 5 種類を使用している．

変数	半角記号
S_1	半角スペース「 」
S_2	セミコロン「;」
S_3	シングルクォーテーション「'」
S_4	右側丸括弧「)」
S_5	左側丸括弧「(」

表 2　攻撃を特徴づける半角記号

これらの半角記号S_1，S_2，S_3，S_4，S_5からなる文字集合から攻撃検出力と正常検出力が最も高い組み合わせを $S = \{S_2, S_2, S_4, S_4\}$と定義し集合 S に属する半角記号が l に出現する総数を$\#S$ としている．この際，攻撃文であることを特徴づける記号の割合は以下の式で定義されている．

$$p_i = \frac{\#S}{l_i} \in [0,1] \quad (1)$$

このp_iを指標として攻撃文字列であるか通常文字列であるかの判定を行う．判定を行うに際して，$h:[0,1] \rightarrow \{0,1\}$を

$$h(p_i) = \begin{cases} 1(p_i > \alpha); \\ 0(p_i < \alpha), \end{cases} \quad (2)$$

と定義し，$h=1$ の時のl_iを攻撃文字列，$h=0$ の時のl_iを通常文字列として判別している．なお，定数αは判別のための閾値である．

この手法の最大の問題点は 5 種類の攻撃を特徴づける文字の中に半角スペースが入っていることが挙げられる．半角スペースは攻撃文字列でも多く使われるが，それ以外にも英文などで区切り文字として多用される記号であり，それ以外の 4 記号についても顔文字やアスキーアート，数式などによく使われる記号であるため，このような記号を含む通常文字列があった場合，攻撃文字列であると誤検知してしまう可能性が極めて高い．

2.2 その２「ブラックリストに依存しない SQL インジェクション攻撃検出法―攻撃の特徴となる半角記号と予約語を用いて―.」[4]

そこで，米内山ら[4]の研究では，小泉ら[3]の提案手法をベースに攻撃文字列の特徴となる半角記号だけでなく，SQL 文の特徴的な予約語を用いて SQL インジェクション攻撃を検知する方法を提案している．攻撃文を特徴づける記号そのものは表 2 と同じ 5 種類を使用し，新たな要素として追加される予約語を以下のように定める．

変数	予約語	変数	予約語	変数	予約語	変数	予約語
w_1	Select	w_4	If	w_7	Where	w_{10}	Set
w_2	Char	w_5	Or	w_8	Insert	w_{11}	Alter
w_3	Create	w_6	From	w_9	Delete	w_{12}	Table

表 3　攻撃を特徴づける予約語

これらの 12 個の単語 w_1，w_2，…，w_{12}について，各単語が入力文字列 l に含まれる総数をそれぞれ$\# w_1$，$\# w_2$，…，$\# w_{12}$と定義し，入力文字列l_iを構成する単語c_iに対する W の割合を以下のように定義する．

$$q_i = \frac{\sum_{a=1}^{W} \#w_a}{c_i} \quad (3)$$

この研究では，攻撃文字列と通常文字列を(2)，(3)式の p_i，q_iの 2 種類の指標を用いて判定する．それぞれの指標で判別する関数として$\mu_\cap$，$\mu_\cup: [0, 1] \rightarrow \{0, 1\}$を

$$\mu_\cap(p_i, q_i) = \begin{cases} 1[(p_i > \alpha) \cap (q_i > \gamma)]; \\ 0(other), \end{cases} \quad (4)$$

$$\mu_\cup(p_i, q_i) = \begin{cases} 1[(p_i > \alpha) \cup (q_i > \gamma)]; \\ 0(other), \end{cases} \quad (5)$$

と定義する．入力文字列が攻撃文であるかどうかは$\mu_\cap$，$\mu_\cup$のどちらかを用いて行い，どちらの場合でもその値が 1 であるときは攻撃文字列，0 であるときは通常文字列だと判定する．なお，定数 α と定数 γはそれぞれ半角記号の割合，予約語の割合に基づく判別を行うための閾値である．この α と γの値を徐々に変化させていったところ，α=0.03, γ=0.19 で関数$\mu_\cap$を用いた時に検知率が最大になったとされている．

この研究において，従来の研究よりも検知率が上昇し，攻撃文字列として誤検知されていた顔文字や URL についても正しく判定できることが確認されている．しかしながら従来研究より大幅に精度を改善することが出来たといってもその検出率は最大でも 85%に満たないという問題を抱えていた．

3. 研究目的

SQL インジェクション攻撃対策，研究に対する前述のような問題点と，さらに既存研究で行われている頻度解析を利用した SQL インジェクション攻撃の検知方法を踏まえ，本研究では，既知の攻撃はもちろん，未知の攻撃も正しく検知することを目的として，SQL インジェクション攻撃に

ⓒ2015 Information Processing Society of Japan

含まれる文字の出現頻度とその関連性の解析による手法を提案する．SQL インジェクション攻撃に含まれる文字同士の関連性をより深く調べ，利用するために，自然言語処理でよく利用されている N-gram 解析を利用した．結果，攻撃文字列，通常文字列，共に高い確率で正しく検知することに成功した．

しかし N-gram 解析のみでは通常文を攻撃文だと誤検知してしまいやすい課題があったため，更なる検知率の上昇と誤検知を減らすことを目的に Skip-gram 解析という手法も活用した．その際，Skip-gram 解析を単に文字単位で行うと N-gram 解析よりも精度が下がってしまうことから，従来研究の 2 つ目として説明した SQL インジェクション攻撃の特徴となる予約語を w_1，w_2，…，w_{12}に置き換えて解析することにする．

なお，N-gram 解析，さらにその発展型の解析法である Skip-gram 解析では N が 3 以下の場合のみ有効であるとされていたが[5]，本研究では各文字片に重みを付けることにより，その精度を格段に上昇させることに成功している．

4. 提案手法

前述の 2 つの既存研究を踏まえ，本研究では，N-gram 解析，Skip-gram 解析とベイズの定理を利用し，入力文字列が攻撃文，通常文のどちらである可能性がより高いかを判定する方法を提案する．

4.1 N-gram 解析

まずは Skip-gram 解析のもとにもなる N-gram 解析について説明する．解析の手順は以下に示すとおりである．

(1) 通常文と攻撃文の訓練データを準備する．
(2) 正常データを N-gram 解析し，得られた文字片ごとの出現回数を数え，正常データに対しそれらの文字列片が出現する確率を P(N)とする
(3) 攻撃データにも(2)と同様の作業を行い，攻撃データに対して N-gram 解析の結果得られた文字列片が出現する確率を P(A)とする．
(4) P(N|S)を入力文字列 S が通常文である確率，P(A|S)を S が攻撃である確率と定義し，以下の式からこれらの値を算出する．

$$P(N|S) = \frac{P(S|N)P(N)}{P(S)} \quad (6)$$

$$P(A|S) = \frac{P(S|A)P(A)}{P(S)} \quad (7)$$

ここで P(S|N)や P(S|A)は 1-gram 解析の場合は単純に内部ベイズと同様の方法で計算し，2-gram 解析，3-gram 解析の場合は，文字列 S を文字列片に分解したあと，その前後の出現頻度をもとに条件付き確率を利用して計算することとする．なお，文字列 S に関する分布 P(S)は，訓練データに依存しすぎてしまうのを防ぐ目的で，事前知識として特徴的な文字片の出現確率を調整するために利用した．

4.2 Skip-gram 解析

次に N-gram 解析を応用した Skip-gram 解析の方法を説明する．なお，Skip-gram は 1-gram 解析では適用出来ないため，Skip-gram を適用する N-gram 解析の N の数値は 2 以上とする．

- N=2 の場合

(1) 通常と攻撃の訓練データを準備する．
(2) 通常データを 1 文字飛ばしで 2-gram 解析し，得られた文字片ごとの出現回数を数え，通常データに対してそれらの文字列片が出現する確率を P(N)とする．
例えば解析する文字列を abc として Skip-gram 解析した文字片のうち，最初の文字が a であるものを取り出すと ac の 1 通りの文字片が挙げられる．
(3) 攻撃データにも(2)と同様の作業を行い，攻撃データに対して 2-Skip-gram 解析の結果得られた文字列片が出現する確率を P(A)とする．
(4) P(N|S)を入力文字列 S が正常である確率，P(A|S)を S が攻撃である確率と定義し，式(6)(7)からこれらの値を算出する．

- N=3 の場合

(1) 正常と攻撃の訓練データを準備する．
(2) 正常データを 1~2 文字飛ばしで 3-gram 解析し，得られた文字片ごとの出現回数を数え，正常データに対しそれらの文字列片が出現する確率を P(N)とする．
例えば解析する文字列を abcdefg として Skip-gram 解析した文字片のうち，最初の文字を a であるものを取り出すと abd，abe，acd，ace，acf，ade，adf，adg の 8 通りの文字片があげられる．
手順(3)，(4)は N=2 の場合と同様．

4.3 攻撃文を特徴づける予約語の利用

Skip-gram を行う際，単に文字単位で解析を行うと N-gram 解析を行っただけの場合よりも結果が良くないことが分かった．そのため，既存研究[4]で利用されている予約語を表 3 に記載されている通りの w_1，w_2，…，w_{12}に置き換えて解析する．

5. 実験結果

提案した攻撃検知手法の有効性を確認するため，初めに 100 個の正常訓練データと 100 個の攻撃訓練データを用意し，P(N)と P(A)の値を算出した．次に，試験用のデータを以下の通り準備し，(6)，(7)式を計算することで試験用データの検知を行った．ここで使用している攻撃データは文献[9]，[10]，[11]から集めたデータを利用し，ランダムで攻撃文を生成するジェネレータを用いて作成している．

ⓒ2015 Information Processing Society of Japan

【通常データ】
Data1：試験用正常データ集合 1（1000 個）
Data2：試験用正常データ集合 2（1000 個）
【攻撃データ】
Data3：試験用攻撃データ集合 3（1000 個）
Data4：試験用攻撃データ集合 4（1000 個）
Data5：試験用攻撃データ集合 5（1000 個）
Data6：試験用攻撃データ集合 6（1000 個）
Data7：試験用攻撃データ集合 7（1000 個）
Data8：試験用攻撃データ集合 8（1000 個）
Data9：試験用攻撃データ集合 9（1000 個）
Data10：試験用攻撃データ集合 10（1000 個）

上記 10 個のデータ集合に対し，以下のパターンの解析を試みた．その結果の平均値を表 4~7 に示す．なお，いずれの表においても単位は%である．

A. テストデータから記号のみを取り出し 1-gram 解析を行う．
B. テストデータから記号と英数字を取り出し 1-gram 解析を行う
C. テストデータから記号のみを取り出し 2-gram 解析を行う．
D. テストデータから記号と英数字を取り出し 2-gram 解析を行う
E. テストデータから記号のみを取り出し 2-Skip-gram 解析を行う（ac 型）．
F. テストデータから記号と英数字を取り出し 2-Skip-gram 解析を行う（ac 型）．
G. テストデータから記号のみを取り出し 3-gram 解析を行う．
H. テストデータから記号と英数字を取り出し 3-gram 解析を行う．
I. テストデータから記号のみを取り出し 3-Skip-gram 解析を行う（ace 型）．
J. テストデータから記号と英数字を取り出し 3-Skip-gram 解析を行う（ace 型）．
K. テストデータから記号のみを取り出し 3-Skip-gram 解析を行う（adg 型）．
L. テストデータから記号と英数字を取り出し 3-Skip-gram 解析を行う（adg 型）．
M. テストデータから記号のみを取り出し 3-Skip-gram 解析を行う（acd 型）．
N. テストデータから記号と英数字を取り出し 3-Skip-gram 解析を行う（acd 型）．
O. テストデータから記号のみを取り出し 3-Skip-gram 解析を行う（ade 型）．
P. テストデータから記号と英数字を取り出し 3-Skip-gram 解析を行う（ade 型）．
Q. テストデータから記号のみを取り出し 3-Skip-gram 解析を行う（adf 型）．
R. テストデータから記号と英数字を取り出し 3-Skip-gram 解析を行う（adf 型）．
S. テストデータから記号のみを取り出し 3-Skip-gram 解析を行う（acf 型）．
T. テストデータから記号と英数字を取り出し 3-Skip-gram 解析を行う（acf 型）．
U. テストデータから記号のみを取り出し 3-Skip-gram 解析を行う（abd 型）．
V. テストデータから記号と英数字を取り出し 3-Skip-gram 解析を行う（abd 型）．
W. テストデータから記号のみを取り出し 3-Skip-gram 解析を行う（abe 型）．
X. テストデータから記号と英数字を取り出し 3-Skip-gram 解析を行う（abe 型）．

解析パターン	検知率	解析パターン	検知率
A	99.97	M	94.41
B	96.85	N	93.69
C	95.01	O	94.32
D	99.67	P	94.51
E	94.38	Q	92.99
F	99.63	R	93.31
G	94.67	S	93.12
H	93.12	T	92.91
I	94.67	U	92.67
J	94.31	V	92.88
K	94	W	93.45
L	93.89	X	92.84

表 4: 予約語を利用しなかった場合の通常文字列の検知率

解析パターン	検知率	解析パターン	検知率
A	98.57	M	93.41
B	96.49	N	93.72
C	96.30	O	94.79
D	96.45	P	94.89
E	96.60	Q	93.30
F	96.41	R	91.06
G	96.49	S	93.03
H	95.32	T	92.40
I	94.66	U	91.68
J	93.13	V	92.01
K	94.89	W	91.76
L	94.01	X	92.24

表 5: 予約語を利用しなかった場合の攻撃文字列の検知率

ⓒ2015 Information Processing Society of Japan

解析パターン	検知率	解析パターン	検知率
B	99.2	N	98.76
D	98.23	P	98.43
F	98.56	R	98.65
H	98.89	T	98.39
J	99.04	V	99.01
L	98.83	X	98.79

表 6: 予約語を利用した場合の通常文字列の検知率

解析パターン	検知率	解析パターン	検知率
B	97.37	N	96.98
D	97.11	P	96.02
F	96.89	R	95.52
H	95.69	T	95.76
J	96.71	V	96.86
L	95.13	X	97.07

表 7: 予約語を利用した場合の攻撃文字列の検知率

なお，ここで扱う予約語はすべてアルファベットであるため，表 6, 7 においてテストデータから記号のみを取り出した解析パターン A，C，E，G，I，K，M，O，Q，S，U，W については予約語なしの場合と変わらないので表から削除してある．

6. 考察と今後の課題

実験の結果，本研究の提案手法を使うことで従来研究よりもさらに高い精度で攻撃文，通常文をともに検知できていることがわかる．従来研究[4]では文字単位の特徴抽出に加えて SQL 文に特有の予約語を攻撃文字列の特徴として用いることで SQL インジェクション攻撃を検知する方法を提案しており，従来研究[3]に比べるとその精度は格段に上昇したとされているが，問題点として検出率が最大でも 85%に満たないことが挙げられていた．本研究で提案した手法では従来研究を応用しつつ, N-gram 解析, Skip-gram 解析などの新しい手法も取り入れることで従来研究の問題点である検知率の低さを十分にカバーできるものであると考える．

実験の結果から，1-gram 解析を行った場合，通常文字列の検知では，記号だけを用いて特徴抽出を行うより，アルファベットを混ぜて特徴抽出を行う方がより確実に検知できるが，攻撃検知では若干検知精度が悪化するという結果が得られた．一方で, 2-gram 解析, 2-Skip-gram 解析, 3-gram 解析，3-Skip-gram 解析を行った場合，通常文字列検知，攻撃文字列検知ともにその検出率が向上することがこの実験では確認することができた．この結果は，SQL インジェクション攻撃を検知するには，文字列に含まれる記号だけを見るよりも，その記号の前後に出現する記号や文字との関連性を用いて攻撃を検知する方がより確実に攻撃検知できる可能性があることを示していると考えられる．

また，単に Skip-gram 解析や N-gram 解析を行うより，予約語を特定の文字で置き換えた上で解析を行った方が大幅に精度の良い検知が行われることがわかる．このことから SQL インジェクション攻撃には SQL 文に特有の予約語が攻撃文字列の特徴として顕著に表れていることもわかる．

本提案法の今後の課題としては，解析の際，訓練データに依存しすぎてしまうのを防ぐ目的で，事前知識として特徴的な文字片の出現確率を調整するために利用した文字列 S に関する分布 P(S)を調整することで更なる検知率の上昇が見込まれるということである．現在の P(S)は事前知識なしで，理想的だと考えられる攻撃文字列訓練データ，通常文字列訓練データについてそれぞれ 1-gram 解析, 2-gram 解析，3-gram 解析を行って求められた文字片ごとの出現頻度と SQL 文字列の特徴となる表 2 の記号や表 3 の予約語に特に重みを付けることで計算している. この P(S)について，さらに最適な値を求めることで，より一層の検知率上昇が見込まれる．

参考文献

1) Takeshi Matsuda, “Feature of SQL Injection Attacks and Zeta Distribution”, FIT2013.
2) 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.
3) 小泉大城，松田健，園田道夫，平澤茂一: 文字集合からの特徴抽出による SQL インジェクション攻撃の自動検出における閾値学習アルゴリズム.
4) 米内山ひかり，中澤真: ブラックリストに依存しない SQL インジェクション攻撃検出法—攻撃の特徴となる半角記号と予約語を用いて—.
5) 伊波靖，安里梓，高良富夫: SVM を利用した WAF の検知方法の提案.
6) 松田健: SQL インジェクション攻撃自動検出支援モデルと予測誤差.
7) 松田健: 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.
8) 園田道夫，松田健，小泉大城，平澤茂一: 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.
9) SQL Injection Cheat Sheet
http://ferruhmavituna.com/sql-injection-cheatsheet-oku/
10) SQL Injection Cheat Sheet
http://michaeldaw.org/sql-injection-cheat-sheet
11) MySQL SQL Injection Cheat Sheet
http://pentestmonkey.net/blog/mysql-sql-injection-cheat-sheet

ⓒ2015 Information Processing Society of Japan