研究指導　中澤 真 准教授

ブラックリストに依存しない SQL インジェクション攻撃の検出法 —攻撃の特徴となる半角記号と予約語を用いて—

米内山ひかり

1. はじめに

近年，Web 上で多くのサービスが利用できるようになった．身近な例として，各 SNS をはじめ，amazon を代表とするインターネットショッピングなどが挙げられる．これらの Web サービスは，アカウントを作れば誰でも利用でき，また，時間や場所を選ぶことなく利用できるため利便性が非常に高く，ユーザ数が飛躍的に伸びている．これは，Web サイトが取り扱う個人情報の増大を意味し，サイト管理者に情報漏えいのリスクが重くのしかかることになる[1]．情報漏えいの原因には内部要因と外部要因の 2 種類があり，その両方について対策を行う必要がある[2]．内部要因での情報漏えいは主に，従業員や外注業者によって行われる．それに対し，外部要因における情報漏えいは，第三者による Web サイトへの不正アクセスによるものである．

Web サイトに不正にアクセスし，登録されている個人情報を盗み取る方法の1つに SQL インジェクション攻撃がある．これは，Web サイト上で入力された文字列について「本来はデータであるはずの文字列が，データベースの問い合わせに使う SQL 文として認識されてしまう」というバグを悪用して行う攻撃手法である．入力フォームに不正な文字列を入力することで，データベースへの不正アクセスを行う[3]．この攻撃は古くから存在し，構文解析や過去の攻撃のブラックリスト化による手法[1]がその対策として広く使われている[4][5]．

しかし SQL インジェクション攻撃は，偽装工作による多様な攻撃が行われており，これらの検出法では先回りでの対応が困難になってきている．この解決策として，園田ら[6]は文字単位での特徴抽出によって攻撃文字列かどうかを判別することで，ブラックリストに依存しない SQL インジェクション攻撃の検出法を提案している．この検出法は半角スペースなどの半角記号を攻撃かどうかの判別基準として用いるため，英文や顔文字の含まれる文章，URL などを攻撃文字列であると誤った判別をしてしまう場合がある．

本研究では園田らの手法の攻撃を特徴づける要素に SQL の予約語を加えることで，これを解決する攻撃検出アルゴリズムを提案し，攻撃検出力と正常検出力の両方の向上を図る．

2. SQLインジェクション攻撃の現状と対策

2.1. インターネット上での個人情報利用の増加

現在のWebサービスはLINEやTwitterなどのソーシャルメディアから，amazonや楽天などのショッピングまで多岐にわたり，多くの消費者がこれらの利便性を享受している．しかし，これらのサービスにはパスワードやメールアドレス，住所，クレジットカード番号など，様々な個人情報が利用されている点に注意する必要がある．この種の情報が第三者の手に渡り，不正利用されてしまうリスクがあるからである．特に企業においては，被害者への謝罪費用や調査費用といったコスト面だけでなく，企業としての信頼やブランドイメージの失墜など，そのダメージは計り知れない．実際，2012年から2013年にかけて発生した，米NASDAQや7-Elevenなど世界大手企業に対するサイバー攻撃では，合計で実に1億6千万件あまりのクレジットカード情報が漏洩し，これら企業の信頼を大きく損なうこととなった[7]．

このような攻撃にはいくつかの方法があるが，中でも大きな脅威となっているものがSQLインジェクション攻撃である[3][4]．本研究ではこの攻撃への対策について論じる．

2.2. SQLインジェクション攻撃の概要

一般的なデータベースでは，SQL という言語を用いてデータの検索や追加・削除などの操作をシステム内で処理する．しかし，Web サイトやデータベースの設定が不適切であったり，セキュリティ上のバグがあったりする場合には，Web ページの入力フォームなどから SQL 文を入力することで，データベースを外部から操作できてしまう可能性がある．このように，通常であれば Web アプリケーションの内部で行われているデータベースとのやりとりを，外部から直接操作することで機密情報へアクセスしたり，データベースを停止させたりする攻撃が SQL インジェクション攻撃である[3]．また，この攻撃ではユーザ認証を行わずに Web サイトへアクセスしたり，Web サイトを改ざんしたりすることも可能である．

SQLインジェクション攻撃は，2005年ごろから多数観測されており，時間の経過とともにその攻撃パターンは複雑で多様なものとなっている[4]．このように新しい攻撃パターンが発見されるたびに，それに対処すべく今日まで様々な対策が行われてきた．その対策について次節で述べる．

2.3. SQLインジェクション攻撃への標準的な対策

根本的なSQLインジェクション対策として，プリペアドステートメントと呼ばれるSQL文のテンプレートなどを用いることで不正な入力でのSQL文の改変をできなくしたり，入力フォームへ入力できる文字列を限定したりする方法が提案されている．これらは，技術的な難易度は高くないものの普及はあまり進んでいない[3][8][9]．

その理由として，Webアプリケーション開発を実際に行う上での特徴がいくつか挙げられる．その1つに，開発を請け負う際の条件がある．Webアプリケーション開

[1] 過去に行われた攻撃のパターンを記録しておき，次回以降に記録されたものと同様のパターンの攻撃を検知する手法．

発は短い納期でかつ低予算で行われることが多いため，改修予算を確保することが難しい．また業界の性質上，人の入れ替わりが激しく，攻撃への対策に関する知見が蓄積されにくい現状がある．暫定的な対策として，ブラックリスト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）があるが，商用のものは高価で導入が難しい．また，非商用のものやフリーウェアのWAFは，新しい攻撃パターンが検出される度にそれを手作業で更新する必要があり，攻撃方法が日進月歩で進化し多様化している現在，WAFによる持続的な対応は難しくなってきている[10].

3. 文字単位での特徴抽出による攻撃検出法

3.1. 先行研究における検出法の概要

従来のSQLインジェクション攻撃対策の問題点に対し，園田ら[6]は攻撃文中に多く含まれる半角記号に着目し，これらの攻撃を特徴づける文字が，入力文字列中に占める割合によって，ブラックリストに依存せずに，新しい攻撃パターンも検知できる高速なアルゴリズムを提案している．

この手法ではまず，フォームに入力された文字列をlとおき，入力文字列全体からなる集合をLで表し，入力文字列の文字列の長さを$|l|$と定義している．また，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判別するため，攻撃を特徴づける半角記号として，表1に示す5種類を使用している．

表1:攻撃を特徴付ける半角記号

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(()

また，これら5種類の半角記号s_1, s_2, s_3, s_4, s_5からなる攻撃を特徴づける文字集合から，攻撃検出力と正常検出力が最も高い組み合わせを$S = \{s_2, s_3, s_4, s_4\}$と定義し，集合$S$に属する半角記号が入力文字列$l$に出現する総数を$\#S$と定義している．このときの攻撃を特徴づける半角記号の占有率は以下の式で定義される．

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

この占有率p_iを指標として攻撃文字列であるか正常文字列であるかの判別を行う．判別を行う関数として，$h : [0,1] \rightarrow \{0,1\}$を

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

と定義し，$h = 1$のときl_iを攻撃文字列，$h = 0$のときl_iを正常文字列と判別している．ここで，定数αは判別を行うための閾値を表している．

3.2. 先行研究における検出法の問題点

園田らの手法では，攻撃を特徴づける文字として表1に示した半角記号5種類を用いているが，この中に含まれる半角スペースは攻撃文字列だけでなく通常の英文においても区切り文字として多く使用されている記号である．また，それ以外の4種類の文字についても，Web上で多用される顔文字やアスキーアート，数式などに多く含まれている．よって，このようなタイプの正常文字列であっても表1に示した文字の占有率が高くなる可能性があり，結果としてこれらの文字列を攻撃であると誤検知する危険性が大きい．

そこで本研究では，攻撃を特徴づけるもう一つの要素としてSQL文の予約語を追加した検出法を提案する．

4. 攻撃の特徴となる半角記号と予約語による攻撃検出法の提案

本研究では，前章でも述べた攻撃検出方法をベースとして，攻撃の特徴として予約語という要素を追加することで，より正確なSQLインジェクション攻撃の検出を目指す．

4.1. 特徴抽出における特徴文字・特徴語句の準備

4.1.1. 攻撃を特徴づける文字および単語集合の設定

本研究では，攻撃を特徴づける文字として表1に示した園田らの手法[6]と同じものを用いる．また，入力文字列lを構成する単語の個数をcと定義する．入力文字列lを構成する単語のうち，攻撃を特徴づける単語をwとし，全てのwからなる単語の集合を$W = \{w_1, w_2, \dots, w_{12}\}$と定義する．$w$については参考文献[11]より，表2に示す計12種類のSQLの予約語を抽出した．なお，これらの単語について，大文字と小文字の区別はしないものとする．

表 2:攻撃を特徴づける単語

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

これら12個の単語$w_1, w_2, \dots, w_{12}$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$\#w_1, \#w_2, \dots, \#w_{12}$で定義し，入力文字列$l_i$を構成する単語の個数$c_i$に対する$W$の占有率を次式で定義する．

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

4.1.2. 攻撃文字列および正常文字列のサンプルの設定

参考文献[3][12][13][14][15][16][17]からSQLインジェクション攻撃のサンプルを収集し，150個の要素からなる攻撃文字列の集合を構成した．この集合の各要素は，SQLインジェクション攻撃で使用される区切り文字やSQL文のコマンドなどで構成される文字列である．この中には，文字コードによる攻撃文字列の生成や少ない文字数での構成による，通常では検知されにくい攻撃文字列も含まれている．

また，一般的なWebの入力フォームを想定した正常文字列のサンプルとして250個のパターンを用意した．これらは誤検知しやすい正常文字列を考えるため，英文かつ半角記号や顔文字が含まれる文字列をTwitterから無作為に抽出したものである．

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3および表4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.

表3:攻撃文字列のサンプル

番号	攻撃文字列
1	SELECT CHAR(0x66)
2	Username: admin'--
3	--
4	if (rs.next()) { }
…	…
149	'OR
150	union select '1', concat(uname‖'-'‖passwd) as…

表 4:正常文字列のサンプル

番号	正常文字列
1	UK says it is not ruling out airstrikes against…
2	If Scotland votes for independence, it may not…
3	When would-be jihadists are being seduced by a…
4	Building & Launching a Business; Part 1…
…	…
249	(sin x)' = cos x
250	http://is.gd/delete

4.2. 攻撃文字列と正常文字列を判別する閾値の決定

本研究では攻撃文字列と正常文字列を, p_iとq_iの2種類の指標を用いて判別する. ここで, それぞれの指標で判別する関数として$\mu_\wedge, \mu_\vee : [0,1] \to \{0,1\}$を

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

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

と定義する. 攻撃か否かの判別は$\mu_\wedge$と$\mu_\vee$のいずれかを用いて行い, いずれの場合でも値が1のときを攻撃文字列, 0のときを正常文字列と判別する. ただし, 定数αと定数γはそれぞれ半角記号の占有率, 予約語の占有率, に基づく判別を行うための閾値とする.

次に, サンプルDからn個の攻撃文字列$\{l_1, l_2, \dots, l_n\}$とm個の正常文字列$\{l_{n+1}, l_{n+2}, \dots, l_{n+m}\}$, 合計$n+m$個の文字列をランダム抽出し, 以下の手順で閾値$\alpha$および$\gamma$を決定する[2]. そのために, 以下の2つの指標を定義する.

$$x_{jg} = \frac{\sum_{i=1}^{n} \mu(p_i, q_i)}{n}, \quad (6)$$

$$y_{jg} = 1 - \frac{\sum_{i=n+1}^{n+m} \mu(p_i, q_i)}{m}, \quad (7)$$

x_{jg}は, 攻撃文字列$\{l_1, l_2, \dots, l_n\}$の中で, 両方の閾値$p_i > \alpha$, $q_i > \gamma$を超えて攻撃文字列と正しく判別された割合を示している.

一方, y_{jg}の値は正常文字列を誤って攻撃文字列と判別する確率であり, どちらの指標も1に近いほど精度の高い検出ができていることを意味する. なお, $l_i (i = 1,2, \dots, n+m)$がそれぞれ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.

最後に文字列のサンプルからR回のランダム抽出によりx_{jg}およびy_{jg}の平均値を算出し, これを1:1の均等割合で平均化した値を

$$z_{jgr} = \left(\frac{\sum_{r=1}^{R} x_{jgr}}{R} + \frac{\sum_{r=1}^{R} y_{jgr}}{R} \right) / 2, \quad (8)$$

と定義する. なお, 本研究では$\sum_{r=1}^{R} x_{jgr} / R$のことを攻撃検出力, $\sum_{r=1}^{R} y_{jgr} / R$を正常検出力, z_{jgr}を総合検出力とよぶ.

4.3. 提案する攻撃検出アルゴリズム

前節で定義した式および指標を用いて, 以下に示すアルゴリズムを提案する.

① p_iによる判別のためのJ個の閾値候補$0 \le \alpha_j \le 1 (j = 1,2, \dots, J)$, q_iによる判別のためのG個の閾値候補$0 \le \gamma_g \le 1 (g = 1,2, \dots, G)$をそれぞれ固定する.

② サンプルDからランダム抽出した文字集合$l_i (l = 1,2, \dots, n+m)$について, すべての$i$に対して$x_{jg}$, y_{jg}を計算し, これをR回繰り返す[3].

③ 各閾値α_j, γ_gに対して, $\mu_\wedge(p_i, q_i)$, $\mu_\vee(p_i, q_i)$それぞれの総合検出力$z_{\wedge jgr}$, $z_{\vee jgr}$を求める.

④ 各閾値において, $z_{\wedge jgr} > z_{\vee jgr}$であれば関数$\mu_\wedge$, $z_{\wedge jgr} \le z_{\vee jgr}$であれば関数$\mu_\vee$を判別に用いることを表5のように決定する. また, z_{jgr}が最大となる閾値α_j, γ_gを決定する.

⑤ ④で決定した閾値α_jおよびγ_gと, 判別を行う関数$\mu_\wedge$または$\mu_\vee$を用いて, 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判別をする.

4.4. 提案法の攻撃検出率のシミュレーションによる評価

提案アルゴリズムの有効性評価のため, 文字列集合のサンプルとして, 攻撃文字列・正常文字列ともに20, 40, …, 100個までをそれぞれ10パターンをランダムに生成した組み合わせを250パターン用意し, Excelのマクロを用いてシミュレーションした. なお, 閾値α_jは0.01から0.15まで, γ_gは0.05から0.20まで変化させている.

提案アルゴリズムのシミュレーション結果について, 各閾値および判別を行う関数の組み合わせを表5に示す. ここでは, 判別を行う際に関数$\mu_\wedge$を用いる場合を「OR」, 関数$\mu_\vee$を用いる場合を「AND」で表している. また, 閾値が$\alpha_j = 0.03$, $\gamma_g = 0.19$で, 関数$\mu_\vee$を用いたときにz_{jgr}は最大値を示した.

表5:閾値の組み合わせ

q_i \ p_i	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1	0.11	0.12	0.13	0.14	0.15
0.05	AND	AND	AND	AND	AND	AND	OR	OR	OR	OR	OR	OR	OR	OR	OR
0.06	AND	AND	AND	AND	AND	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR
0.07	AND	AND	AND	AND	AND	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR
0.08	AND	AND	AND	AND	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR
0.09	AND	AND	AND	AND	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR
0.1	AND	AND	AND	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR
0.11	AND	AND	AND	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR
0.12	AND	AND	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR
0.13	AND	AND	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR
0.14	AND	AND	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR
0.15	AND	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR
0.16	AND	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR
0.17	AND	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR
0.18	AND	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR
0.19	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR
0.2	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR

[2] シミュレーション実験では n=150, m=250 である.

[3] シミュレーション実験では R=250 である.

次に，園田らの手法である半角記号のみで判別を行った場合，攻撃を特徴づける予約語のみで判別を行った場合，本研究の提案方法による場合の総合検出力をそれぞれシミュレーションし，本研究の有効性を示す．

先行研究と本研究との比較ではγ_gを固定してα_jを変化させ，予約語のみの判別手法と本研究との比較では，α_jを固定し，γ_gを変化させた場合の総合検出力の変化をシミュレーションしている．なお，それぞれの閾値を固定する場合は，総合検出力が最大値を示したケースの$\alpha_j = 0.03$, $\gamma_g = 0.19$としている．

閾値αの変化による先行研究と本研究の総合検出力の比較を図1に，閾値γの変化による予約語のみの単純判別法と本研究の総合検出力の比較を図2に示す．

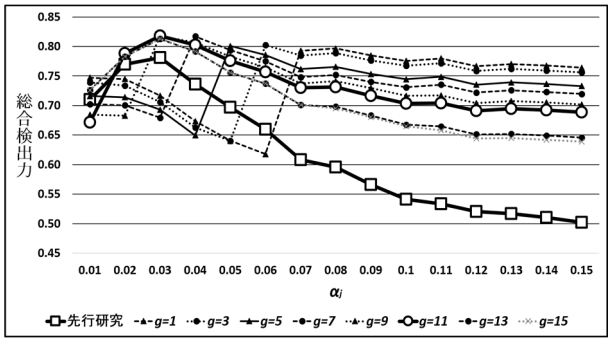

図1:先行研究と本研究との比較

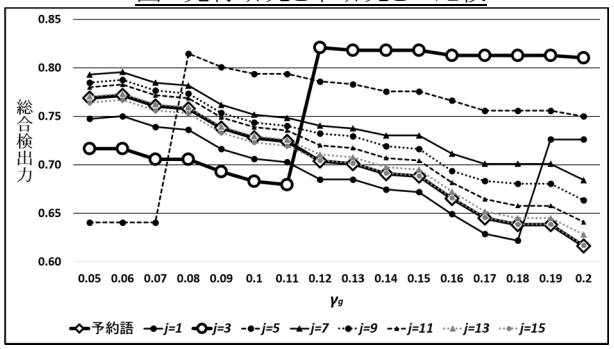

図2:予約語のみでの判別と本研究との比較

図1より，閾値を変化させた場合の総合検出力について，先行研究では急激に低下していることがわかる．それに対し，本研究ではいずれの場合もゆるやかに低下しており，比較的安定した検出を行っている．

4.5. 考察

シミュレーションの結果，本研究が従来手法よりも高い総合検出力を維持できていることが明らかになった．このように，攻撃文字列の特徴づけを行う要素としてSQLの予約語を含めることで，半角記号の占有率のみで行う検出法では見落とされやすい攻撃文字列も正しく検出できることが確認された．また，攻撃と間違われやすい顔文字やURLについても正しく判別できることが確認された．

判別のための計算量については，入力文字列を過去に記録された攻撃パターンすべてと照合するブラックリスト方式に対し，フォームへの入力文字列に対する半角記号と予約語の占有率をそれぞれ計算するのみであり，これは園田らの手法と同等の高速処理が可能である．

5. むすび

本研究では，文字単位の特徴抽出に加え，SQLの予約語を攻撃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出するアルゴリズムを提案し，抽出・生成したデータを用いて評価を行った．提案法では，閾値の決定だけでなく，2つの閾値の組み合わせにも着目し，より正確な攻撃および正常文字列の検出を実現している．

提案法の課題は，攻撃文字列と正常文字列を正しく判別する精度である．前述したとおり，本研究では園田らの手法に比べ大幅に向上させることができたが，その検出率は最大でも85％に満たない．よって，攻撃の特徴となる予約語について，どの予約語を用いればより正確な判別が可能になるのかを今後検証する必要がある．

参考文献・URL

[1] NRIセキュアテクノロジーズ株式会社,“クラウド時代の情報セキュリティ,”日経BP社,2010年.
[2] セコム,セコムトラストシステムズが考える情報漏洩対策,http://www.secomtrust.net/infomeasure/rouei/column1.html
[3] 山崎洋一,WAFがよく検知するWeb攻撃は?,ITpro by日経コンピュータ,http://itpro.nikkeibp.co.jp/atcl/news/14/092601091/
[4] 谷口隼裕,SQLインジェクション対策について,IPA, http://www.ipa.go.jp/files/000024396.pdf
[5] 神崎洋治・西井美鷹,体系的に学ぶインターネットセキュリティ,日経BPソフトプレス,2008年.
[6] 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,”研究報告コンピュータセキュリティ(CSEC),52巻,49号,pp.1-7,2011年3月.
[7] 鈴木聖子, 世界の大手企業に不正アクセスした犯人,1億6000万件のクレジットカード情報を盗む,ITmedia, http://www.itmedia.co.jp/enterprise/articles/1307/26/news032.html
[8] PHP0,サニタイジングとエスケープの違い「セキュリティ(調査結果)」カテゴリー,http://php0.e1blue.net/php/status/79
[9] 徳丸浩,ホワイトリスト方式の優位は神話, http://www.tokumaru.org/d/20080716.html
[10] 楽天理,WAF製品選びの鉄則ーアプライアンスか?ソフトウェアか? SaaS型も登場!,business network, http://businessnetwork.jp/Detail/tabid/65/artid/391/Default.aspx
[11] Microsoft Developer Network,予約済みキーワード, http://msdn.microsoft.com/ja-jp/library/ms189822.aspx
[12] JVN iPedia 脆弱性対策情報データベース,SQLインジェクション,http://jvndb.jvn.jp/ja/cwe/CWE-89.html
[13] PHP,SQLインジェクションとは, http://php.net/manual/ja/security.database.sql-injection.php
[14] はるぷ,不正なJsonデータによるSQLインジェクションへの対策について(Json.pm＋SQLクエリビルダー), http://developers.mobage.jp/blog/2014/7/3/jsonsql-injection
[15] 長谷川武,脆弱なWebアプリケーション,セントラル・コンピュータ・サービス,Think’IT, http://thinkit.co.jp/free/tech/7/5/1.html
[16] 徳丸浩,SQLインジェクションゴルフ, http://blog.tokumaru.org/2013/06/sql-injection-golf-3-letters-bypass-login-authentication.html
[17] 白夜書房,SQL Injection Cheat Sheet, Document Version 1.4,http://www.byakuya-shobo.co.jp/hj/moh/sqlinjectioncheatsheet.html