A Learning Algorithm of Threshold Value on the Automatic Detection of SQL Injection Attack

Daiki Koizumi∗, Takeshi Matsuda∗, Michio Sonoda∗, and Shigeichi Hirasawa∗
∗ Faculty of Information Technology and Business, Cyber University, Tokyo, Japan

Abstract—The SQL injection attack causes very serious problem to web applications that involve database including personal data. To detect the SQL injection attack, the parsing and the blacklist built based on the known attacks have been widely used. Those approaches, however, have some problems in terms of the size of list or calculation costs as the number of attacks increases. For these problems, the authors have previously proposed a simple automatic algorithm to detect SQL injection attack. This algorithm requires to calculate the ratio of suspicious characters contained in an input sequence. This rate is compared with a known real-valued threshold. This paper proposes the learning algorithm to choose the real-valued threshold from training data sets. Furthermore, some criteria will be considered and their performances will also be examined.

Keywords: SQL Injection Attack, Automatic Detection, Web Application, Learning Algorithm, Pattern Recognition

1. Introduction

The SQL injection attack was firstly reported in 1999 [2]. It consists of insertion or “injection” of a SQL query via the input data from the client to the application [3]. It causes very serious problem to web applications that involve database including personal data. To detect the SQL injection attack, the parsing and the blacklist built based on the existed attack have been widely used. However, the conventional methods based on blacklist built approaches have confronted at least two problems. The first is the huge cost of updating large blacklist built, the second is the calculation cost for automatic attack detection with large blacklist built.

For these two problems, the authors have already proposed the online algorithm for automatic detection of SQL injection attack [4]. This algorithm requires to calculate the contained rate of suspicious characters with input sequence. This rate is compared with a known real-valued threshold. In the authors’ previous research [4], this threshold was empirically determined as a known constant. However, in general, this threshold is an unknown value and should be learned from a given training set of data. This paper proposes a learning algorithm to choose a real-valued threshold. Furthermore, some criteria will be considered and their performances will be also examined.

The rest of this paper is organized as the followings. Section 2 gives some definitions and automatic detection algorithm of SQL injection attack which was previously proposed where its threshold value is defined as a known constant. Section 3 proposes the learning algorithm of the threshold value with some formulations. Section 4 shows some leaning examples with artificial data. Section 5 gives their discussions. Finally, Section 6 concludes this paper.


2.1 Preliminary

Suppose that $l$ is an input sequence through web application to the SQL database. Note that each input $l$ has a label either attack or normal. The purpose of automatic SQL injection attack detection is to estimate the label of $l$ correctly. Our proposed algorithm requires some known finite characters $s_1, s_{ii}, s_{iii}, \ldots$. These are suspicious characters contained in SQL injection attack inputs such as space, semicolon, single-quotation, left and right round brackets, and so on. Furthermore, let $S_k$, $k = 1, 2, \cdots, m$ be power sets of known characters $s_i, s_{ii}, s_{iii}, \cdots$ where they are defined as sets of characters except for the empty set. If $s_i$ and $s_{ii}$ are defined as space and semicolon, then three power sets of known characters can be $S_1 = \{s_i\}, S_2 = \{s_{ii}\},$ and $S_3 = \{s_i, s_{ii}\}$.

2.2 The Automatic Detection Algorithm

With the above definitions, the automatic detection algorithm was proposed [4]. Furthermore, the theoretical performance was analyzed in terms of statistical prediction problem [1]. The following is the brief description of the automatic detection algorithm.

1) Setting up known values
a) Choose a set of characters $S_k$.

b) Set a threshold as a real value $\alpha \in [0, 1]$.

c) Calculating the content rate of suspicious characters $x_{k,l}$ which is defined as,

$$x_{k,l} = \frac{\#S_k}{|l|},$$

(1)

where $\#S_k$ denotes the size of $S_k$.

3) Automatic detection

Determine each input’s label (normal or attack input) by the following function $d(x_{k,l}, \alpha)$:

$$d(x_{k,l}, \alpha) = \begin{cases} 
0 & \text{if } x_{k,l} \leq \alpha; \\
1 & \text{if } x_{k,l} > \alpha.
\end{cases}$$

(2)

Eq. (2) means that if detection result is normal, then its value is zero, otherwise the result is attack and its value is one.

**Example 2.1 (Automatic Detection of Attack Input):**

Let $l$ = “DROP sampletable;:- -” be an attack input where its length $|l| = 19$. Suppose that a character set $S_{13}$ contains space, semicolon, and left round brackets. Furthermore, the threshold value $\alpha$ is set to 0.10.

Since the input $l$ contains one space and one semicolon among characters in $S_{13}$, a numerator in Eq. (1) becomes $\#S_{13} = 2$. Therefore, according to Eq. (1),

$$x_{13,l} = \frac{2}{19} = 0.1052 \ldots$$

With the above $x_{13,l}$, the detection result by Eq. (2) becomes the following:

$$d(x_{13,l}, \alpha) = d(0.1052, 0.10) = 1.$$  

Thus $l$ is detected as an attack input.

### 2.3 Performance Evaluation with Artificial Data

For evaluation of the above algorithm, the artificial data was composed [4]. Those data cover the typical types of SQL injection attack input as well as normal input among common web forms. The number of types of attack inputs was 624, on the other hand, that of normal inputs was 234. Those data were converted to the fields of single and multibyte characters, wiki, emoticon etc. These types were assumed to be input as IDs, passwords, names, and addresses etc. [4].

If the real operating situation is considered, the label of each input (either normal or attack) is unknown. Therefore, the mixture data of both labels were used for simulations in evaluations [4]. Let $0 \leq P_N \leq 1$ be the correct detection rate for normal input and let $0 \leq P_A \leq 1$ be the correct detection rate for attack input. Furthermore, if the ratio of the number of the normal input against attack input is $0 \leq \beta \leq 1$, the total detecting rate $0 \leq \mu \leq 1$ can be calculated by the following:

$$\mu(S_k, \alpha, \beta) = \beta P_N + (1 - \beta) P_A.$$  

(3)

For evaluation, the value of $\alpha$ was empirically chosen as a constant. On the other hand, various values of $\beta$ were taken with its interval $0 \leq \beta \leq 1$. For the remained $S_k$, the following objective function was assumed to chose the optimal character set of $S^*$.

$$S^* = \arg \max_{S_k} [\mu(S_k, \alpha, \beta)].$$

(4)

With the above Eq. (4), the sensitive analysis was considered for discussions [4].

### 3. The Proposed Threshold Learning Algorithm

#### 3.1 Formulation and Criterion

In general, the threshold value is unknown and should be learned from real observed data. If $S^*$ has been already determined and the candidates of $\alpha_j$, $j = 1, 2, \ldots, N$ have been obtained, then the following can be defined as similar form of Eq. (4).

$$\alpha^* = \arg \max_{\alpha_j} [\mu(S_k, \alpha_j, \beta)].$$

(5)

Since the label of input $l$ is unknown at the real operation, the value of $\beta$, which is the weight of normal input against attack input is also unknown. Therefore, Eq. (4) and (5) can be achieved with several criteria. One of them is that assuming the probability distribution of $p(\beta)$ to take the expectation of $\mu(S_k, \alpha_j, \beta)$ with respect to $\beta$. For numerical approximation, suppose $\beta_m$, $m = 1, 2, \ldots, M$ is sampled on the interval $0 \leq \beta \leq 1$. Then, such criterion can be formulated as the following:

$$\{\alpha^{**}, S^{**}\} = \arg \max_{\alpha_j} \max_{S_k} \left[ \sum_{m=1}^{M} p(\beta_m) \mu(S_k, \alpha_j, \beta_m) \right].$$

(6)

Note that $\alpha^{**}, S^{**}$ maximize the expected total detecting rate $\mu(S_k, \alpha_j)$ in Eq. (6).

For the other criteria, both the expected total detecting rate and the absolute value of slope of regression line can be considered which is more
restrictive. This criterion also takes into account the stability of \(\mu (S_k, \alpha_j)\) with respect to \(\beta\).

\[
\{\alpha^{**}, S^{***}\} = \arg \max_{\alpha_j} \max_{S_k} \left[ \sum_{m=1}^{M} p(\beta_m) \mu (S_k, \alpha_j, \beta_m) \right] - \frac{1}{M} \left( \sum_{m=1}^{M} \beta_m^2 \right) \sum_{m=1}^{M} \beta_m \mu (S_k, \alpha_j, \beta_m) \left( \sum_{m=1}^{M} \mu (S_k, \alpha_j, \beta_m) \right).
\]

(7)

Note that \(\alpha^{**}, S^{***}\) maximize the sum of the expected total detecting rate and the slope of regression line. In Eq. (7), the second, third, and fourth terms on the right hand side express the absolute value of slope of the regression line \(\mu (S_k, \alpha_j)\) where \(\beta\) is its domain.

3.2 The Proposed Algorithm

With training data set that contain pairs of input sequence \(l\) and its label, the following learning algorithm of threshold value \(\alpha^{**}\) or \(\alpha^{***}\) would be proposed.

1) With various candidate pairs of \(S_k\) and \(\alpha_j\), execute automatic detection algorithm with Eq. (1) and (2).

2) Calculate the total detecting rate \(\mu (S_k, \alpha_j, \beta)\) with \(P_N\), \(P_A\), and \(0 \leq \beta \leq 1\).

3) Taking \(\beta_m\) for numerical approximation to choose the optimal set of \(S^{**}\) and \(\alpha^{**}\) by Eq. (6)(or \(S^{***}\) and \(\alpha^{***}\) by Eq. (7)).

4. Evaluations of the Proposed Algorithm

4.1 Conditions

For evaluations, the artificial data mentioned in subsection 2.3 was used. The number of types of attack inputs is 624, those of normal inputs is 234 where the data cover the single and multibytes characters, wiki, emoticon etc. Note that the data was assumed IDs, passwords, names, and addresses etc. For known finite characters, the five characters were chosen as Table 1. These characters are same as our previous simulations [4].

With the above five characters, the following twenty six power sets can be defined as,

Table 1: Known Characters

<table>
<thead>
<tr>
<th>Name</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>(s_i)</td>
<td>Space</td>
</tr>
<tr>
<td>(s_{si})</td>
<td>Semicolon (:)</td>
</tr>
<tr>
<td>(s_{sii})</td>
<td>Single Quotation ('')</td>
</tr>
<tr>
<td>(s_{sv})</td>
<td>Right Parenthesis (])</td>
</tr>
<tr>
<td>(s_{sl})</td>
<td>Left Parenthesis([)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>(s_1)</td>
<td>(s_{sii}, s_{ii})</td>
</tr>
<tr>
<td>(s_2)</td>
<td>(s_{si}, s_{sii})</td>
</tr>
<tr>
<td>(s_3)</td>
<td>(s_{si}, s_{sii})</td>
</tr>
<tr>
<td>(s_4)</td>
<td>(s_{si}, s_{sii}, s_{sv})</td>
</tr>
<tr>
<td>(s_5)</td>
<td>(s_{sii}, s_{sv}, s_{sii})</td>
</tr>
<tr>
<td>(s_6)</td>
<td>(s_{sii}, s_{sv}, s_{sii})</td>
</tr>
<tr>
<td>(s_7)</td>
<td>(s_{sii}, s_{sv}, s_{sii})</td>
</tr>
<tr>
<td>(s_8)</td>
<td>(s_{sii}, s_{sv}, s_{sii})</td>
</tr>
<tr>
<td>(s_9)</td>
<td>(s_{sii}, s_{sv}, s_{sii})</td>
</tr>
<tr>
<td>(S_1)</td>
<td>(s_{sii}, s_{sii})</td>
</tr>
<tr>
<td>(S_2)</td>
<td>(s_{sii}, s_{sii})</td>
</tr>
<tr>
<td>(S_3)</td>
<td>(s_{sii}, s_{sii})</td>
</tr>
<tr>
<td>(S_4)</td>
<td>(s_{sii}, s_{sii}, s_{sii})</td>
</tr>
<tr>
<td>(S_5)</td>
<td>(s_{sii}, s_{sii}, s_{sii})</td>
</tr>
<tr>
<td>(S_6)</td>
<td>(s_{sii}, s_{sii}, s_{sii})</td>
</tr>
<tr>
<td>(S_7)</td>
<td>(s_{sii}, s_{sii}, s_{sii})</td>
</tr>
<tr>
<td>(S_8)</td>
<td>(s_{sii}, s_{sii}, s_{sii})</td>
</tr>
<tr>
<td>(S_9)</td>
<td>(s_{sii}, s_{sii}, s_{sii})</td>
</tr>
</tbody>
</table>

4.2 Simulations

1) Simulation 1

Choose five pairs of \(\{\alpha^{**}, S^{**}\}\) in descending order according to the criteria in Eq. (6).

2) Simulation 2

Choose five pairs of \(\{\alpha^{***}, S^{***}\}\) in descending order according to the criteria in Eq. (7).

4.3 Results

Table 2 and 3 were obtained for Simulation 1 and 2, respectively.

Table 2: Result of Simulation 1

<table>
<thead>
<tr>
<th>Rank</th>
<th>(S^{<strong>}, \alpha^{</strong>})</th>
<th>Value in Eq. (6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>(S_{22}, \alpha = 0.08)</td>
<td>0.9170</td>
</tr>
<tr>
<td>2nd</td>
<td>(S_{12}, \alpha = 0.08)</td>
<td>0.9154</td>
</tr>
<tr>
<td>3rd</td>
<td>(S_{22}, \alpha = 0.09)</td>
<td>0.9147</td>
</tr>
<tr>
<td>4th</td>
<td>(S_{21}, \alpha = 0.08)</td>
<td>0.9142</td>
</tr>
<tr>
<td>5th</td>
<td>(S_{14}, \alpha = 0.02)</td>
<td>0.9135</td>
</tr>
</tbody>
</table>

Table 3: Result of Simulation 2

<table>
<thead>
<tr>
<th>Rank</th>
<th>(S^{<em><strong>}, \alpha^{</strong></em>})</th>
<th>Value in Eq. (7)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>(S_{12}, \alpha = 0.09)</td>
<td>0.9032</td>
</tr>
<tr>
<td>2nd</td>
<td>(S_{12}, \alpha = 0.08)</td>
<td>0.8994</td>
</tr>
<tr>
<td>3rd</td>
<td>(S_{22}, \alpha = 0.11)</td>
<td>0.8930</td>
</tr>
<tr>
<td>4th</td>
<td>(S_{21}, \alpha = 0.10)</td>
<td>0.8903</td>
</tr>
<tr>
<td>5th</td>
<td>(S_{21}, \alpha = 0.11)</td>
<td>0.8879</td>
</tr>
</tbody>
</table>
5. Discussions

From Table 2 in Simulation 1, we can see that the pair $S_{22}$ and $\alpha = 0.08$ maximizes the expected total detecting rate $\mu(S_k, \alpha_j)$ in Eq. (6). In our previous research [4], the pair $S_{12}, \alpha = 0.08$ was empirically chosen. According to Table 2, relatively superior pair was discovered with the criterion in Eq. (6). Figure 1 shows the plot of the top three pairs of $S^{**}$ and $\alpha^{**}$ where the vertical axis is the value of $\mu$ and the horizontal axis is $0 \leq \beta \leq 1$. According to Eq. (6), the superior $\mu$ of the pair $S_{22}$ and $\alpha = 0.08$ can be observed comparing to the empirically chosen pair $S_{12}$ and $\alpha = 0.08$ in Figure 1. In Figure 1, the pair $S_{12}$ and $\alpha = 0.08$ gives the most flattest line among three lines, however, the other two pairs give the relatively superior values according to Eq. (6).

From Table 3 in Simulation 2, $S_{12}$ and $\alpha = 0.09$ are obtained as the optimal pair according to Eq. (7). The second is the pair of $S_{12}$ and $\alpha = 0.08$. Figure 2 shows the same sort of plot as previous Figure 1. Since Eq. (7) emphasizes the absolute value of the slope, more flatter line can be chosen. In this criterion, $S_{12}$, which contains three characters, was superior to the others, whereas $S_{22}$ was superior to them in the criteria Eq. (6).

Figure 3 shows the effect of various values of $\alpha$ in $S_{12}$. From Figure 3, the more the value of $\alpha$ increases, the more larger the value of slope of the line becomes. Since Eq. (7) gives the penalty of the larger value of the slope, $\alpha = 0.09$ is more likely to be chosen.

Figure 4 shows the effect of various values of $\alpha$ in $S_{21}$. Figure 4 also shows that the more the value of $\alpha$ increases, the more the value of slope of the line becomes larger. But the increasing degree of the slope in $S_{21}$ is relatively larger than that of $S_{12}$. This result can be interpreted as the effect of the character of Right Parenthesis which is the only contained character in $S_{21}$.

Figure 5 shows the effect of various sets among $S_{12}, S_{21},$ and $S_{24}$ where those thresholds are the constant $\alpha = 0.08$. From Figure 5, the detecting performances of $S_{12}$ and $S_{21}$ are similar, however, that of $S_{24}$ is the relatively poor. It can be observed that $S_{24}$ is the only set which does not contain Semicolon.

6. Conclusions

This paper proposed the learning algorithm to choose the real-valued threshold for automatic detection of SQL injection attack. Furthermore, some
learning criteria were considered and their performances were also examined with artificial data. As a result, the certain effectiveness was observed with the proposed algorithm and thus seeking the unknown threshold value can be possible with training sets of data.

For future research, predictive performance should be examined with unknown data sets. Furthermore, the detecting performance with the real SQL injection data should also be considered.

**Acknowledgments**

This research is partially supported by the Grant-in-Aid for Scientific Research (C) No.23501178 of the Japan Society for the Promotion of Science (JSPS).

This research is partially supported by the Grant-in-Aid for Scientific Research (B) No.23740094 of the Japan Society for the Promotion of Science (JSPS).

This research is partially supported by the 44uh Kurata Grants of the Kurata Memorial Hitachi Science and Technology Foundation.

**References**


