WEB APPLICATION VULNERABILITY DETECTION BASED ON REINFORCEMENT LEARNING

Wang Xin, Wei Gengyu, Yang Yixian

1. Key Laboratory of network and information attack & defence technology of MOE, Beijing University of Posts and Telecommunications, Beijing, China, wx.bupt@gmail.com
2. National Engineering Laboratory for Disaster Backup and Recovery, Beijing University of Posts and Telecommunications, Beijing, China

Abstract

To solve the problem of low crawling yield and low detection efficiency in web applications security detection, we propose a web application security vulnerability detection method based on Q-learning. We present a strategy of form focused crawling (QLC) which uses Q-learning algorithm to increase the crawling yield and detection efficiency. In the learning algorithm, we present the method of combining immediate rewards and future rewards to evaluate and optimize the learning rules. Simulating web attacking and analyzing the data of response are used to detect security vulnerabilities, and rich attacking vectors ensure the improvement of detection accuracy. Finally, through effective training of the reinforcement learning the rules, a series of experimental results verify the effectiveness of the method we proposed in this paper.

Keywords: Web security vulnerability, Reinforcement learning, Q-learning, Web detection

1. Introduction

With the development of web technology and the important role of web application, the web security is becoming critical. Web application vulnerability detection based on black-box is the main technology [1–6,14]. With the applying of black-box detection technology, vulnerabilities are detected and the security of web applications is improved effectively, yet it remains problems [14, 15] in some areas. First, the index crawler strategy is used in the traditional black-box web application detection to obtain detect target's basic information [1–6,14,15], but this crawling strategy is blindness in a certain degree, and then the efficiency of detection goes lower[16–19]. Secondly, in the web application, only at the location of the user interaction or form existing place, possible security vulnerabilities could take place[16,14,15], but the traditional detection method lack of such pertinence, and leading to a lower crawling yield[1–6,14,15].

Web crawler is essential in black-box web security detection, the efficiency of which could directly affect the efficiency of detection system. The index crawler [1–6] and topic focus crawler [15,16], which are widely used in traditional detection system, lack form crawling pertinence and have low crawling yield, then it decrease the detection efficiency. Therefore, in web security vulnerability detection, finding effective form crawling methods to increase crawling yield and detection efficiency has become the key point of the study.

In this paper, we propose a web security vulnerability detection method based on reinforcement learning, and design a form focused crawler in which firstly employs Q-learning algorithm. Considering the characteristics of the relationship between forms and links, we present to a method to combine immediately rewards with future rewards in Q-learning algorithm. The process of detection is divided into two steps, which is exploration and exploitation. In exploration, crawler based on Q-learning algorithm is trained and the Q-value table is established. In the exploitation process, the crawler according to the learning results, the integrated Q-value is calculated and the link is selective required according to the integrated Q-value. In this step, the forms are extracted and the learning rules are optimized. Detecting web security vulnerabilities with reinforcement learning based form focus crawler, achieve the purpose of fewer queries more entry points. The method we proposed makes the yield and the efficiency improved 200%–400% more. A series of experiments verified the method, and better practical results are obtained.
The rest of the paper is constructed as follows. In section 2 we describe the related works. In section 3 we present the strategy and the architecture of reinforcement learning based form focus crawler, and then introduce the detecting method of XSS and SQL injection vulnerabilities. Section 4 describes the architecture of detection system we have implemented, and analyzed the experimental results. Finally we conclude this paper and discuss our future works.

2. Related Works

There are some study works on web vulnerability detection and many tools have been developed and used, for example, AppScan [4], WebInspect [1], WVS [2], WebScarab [3], WebRavor [5], etc. All of them use index crawler to crawl web pages and use attacking queries to test vulnerabilities. They are helpful to certain extent in securing web applications, but the low crawling yield and detection efficiency can’t be ignored[14,15,17–19].

In [6], a tool called WAVES has used learning strategy to fill simple web forms (e.g. combox, checkbox, etc), which can gain some of the web pages behind the forms. However, WAVES also crawl all pages not only crawl form related pages, so the crawling yield still keeps higher.

Reinforcement learning [11] and Q-learn algorithm [8–10] have been used in topic focus crawling [12,20,21], it has optimized the crawling efficiency, but the algorithm only used to calculate the relevance between pages and object topics, is does not pay any attention to forms in pages. In addition, the evaluation of reward in [12,20,21] is simple, that ignore the effect between immediate reward and future reward.

3. Web Application Vulnerabilities Detection Based On Reinforcement Learning

In this section, we presents a web application security vulnerability detection method based on reinforcement learning, and design a form focus crawling strategy employing Q-learning algorithm. Through estimating and calculating the integrated Q-value, the form focus crawler could crawl along the direct or indirect form page links. This approach makes the detection concern on possible vulnerabilities entry points (EP) more directly. Simulating web attacking and analyzing the data of response are used to detect security vulnerabilities, and rich attacking vectors ensure the improvement of detection accuracy.

3.1. Reinforcement Learning

Reinforcement learning [8,11] is an area of machine learning in computer science, concerned with how an agent ought to take actions in an environment so as to maximize some notion of cumulative reward.

Q-learning[8,9,10] is a reinforcement learning technique that works by learning an action-value function Q(s, a) that gives the expected utility of taking a given action in a given state and following a fixed policy thereafter. The problem model consists of an agent, states S and a number of actions per state A. By performing an action a, where a∈A, the agent can move from state to state. Each state provides the agent a reward or punishment. The goal of the agent is to maximize its total reward. It does this by learning which action is optimal for each state. Action-value function is described as follow:

\[
Q'(s,a) = R(s,a) + \gamma \sum_{s',a'}P(s,a,s') \max_a Q'(s',a')
\]

Where Q *(s,a) is the best reward fed back by the action a∈A at the state s∈S, the discount factor γ is such that 0≤γ<1.

We define V *(s) is the most optimal function at the station s∈S, so

\[
V^*(s) = \max_a Q^*(s,a)
\]

That is the most optimal strategy is
Before learning has started, $Q$ returns a fixed value, chosen by the designer. Then, each time the agent is given a reward (the state has changed) new values are calculated for each combination of a state $s$ from $S$, and action $a$ from $A$. The core of the algorithm is a simple value iteration update. It assumes the old value and makes a correction based on the new information.

Q-learning iterative formula is shown as follows:

$$Q(s,a) = Q(s,a) + \alpha (r + \gamma \max_a Q(s,a') - Q(s,a)) \quad (4)$$

During initialization, Q-learning algorithm initialize each of the $Q(s,a)$ values (in order to calculate simple, mostly set to 0), then select the biggest $Q$ value according to greedy strategy, and gain the training rules <$s, a, r, s'>$ and through the Q-Learning iterative formula, the actual iterative value is obtained. When it reaches the target state, the iterative process ends. And then continue to iterate from the initial state until the end of the learning process. If every state-action pair is visited infinitely, and the learning rate $\alpha$ decay suitable, then the $Q$ value will eventually converge to $Q^*$, the convergence has been proved.

3.2. Strategy Of Form Focus Crawler Based On Q-Learning

In this section, we introduce a form focus crawler strategy based on Q-learning algorithm (Q-learning Crawler, referred to as QLC) strategy. With this strategy, web security detection will focus on entry point (EP) of web application directly, where possible vulnerability may exist. Applying this method, the detection efficiency and the yield improved.

3.2.1. Training of Q-learning

In web application, not all of the Links point to the relevant page that contains EP, in contrast, EP distribute extremely sparsely in the Web application. Therefore, we divide the links in web application into two categories:

- **Direct EP Link** (DEPL): the Link point to the relevant page that contains EP
- **Indirect EP Link** (IDEPL): the Link point to the relevant page that does not contains EP, but it contain DEPL or IDEPL. If IDEPL finally reached to DEPL by skipping $n$ ($n > 1$) times, we call the IDEPL for the n-IDEPL.

In the QLC, how to estimate and evaluate the value of links is the key point to determine the strategy of form focus crawling. Immediate reward and future reward are the two evaluating methods. The former strategy is more focused on resources exploration, while the latter strategy is more advantage in resource exploitation.

In order to make QLC to concern with the form relevant pages more, and to improve the accuracy of the estimate of link value, we combine immediate reward and future reward, and calculate the integrated $Q$-value. QLC are divided into training and searching phase. In the training phase, the evaluation method based on the strategy of immediate reward is employed. Through which, the URLs with maximum training reward value are obtained and the $Q$-value table is updated. In the searching phase, the future reward based evaluation method is used. With this method, the form relevant URLs are found as much as possible to find the relevant page URL, and the comprehensiveness of QLC could be improve.

In QLC, we define:

- **Reward**: the static or dynamic web page
- **Action**: the following hyperlink
- **Station**: web page with form

When the discount factor $\gamma = 0$, we call formula (1) the IR (Immediate Reward), that the delay reward does not affect the present, that is

$$Q^i(s,a) = R(s,a) \quad (5)$$

When the discount factor $0 < \gamma < 1$, we call formula (1) the FR (Future Reward), that means the following pages affect the current page with the $\gamma$ discount factor.
In the initialization of training, given a set of links which point to the page that contains typical forms, through training, the features of the links are learned and the Q-value table is established. The purpose of training is as similar as the general reinforcement learning with the knowledge of the transfer function \( T \) and reward function \( R \). Then QLC crawls pages and extracts forms basing on the relevance between link and form with the rule of Q-learning algorithm.

It estimates the future reward value which is expressed by the Q-value according to current state. As mentioned above, QLC is divided into two stages of training and searching. During the training stage, reinforcement learning calculates the Q-value of each form related link and divided them into several classes, and establish Q-value table. In the searching stage, integrated Q-value is calculated according to the URL text and the established Q-value table, and the relevance between link and form is determined.

The training stage of QLC is shown in figure 1.

![Figure 1. The training stage of QLC](image)

With the construction of metadata, a set of amount of DEPLs is established. The queries of these DEPLs are not limited by the login authentication. According to the function of form, we divide forms into four categories: search query class, login authentication class, information submitted classes and other type class. By using back draw crawler[21] crawl on the metadata, get the n-level parent pages of metadata, those are n-IDEPLs, and then classified them according to the level of the page. The feature (such as URL, anchor, text around the link and so on) of these parent links are extracted, which are used to calculate the immediate reward value and integrated Q-value. And then the Q-value table is updated.

3.2.2. Integrated Q-Value

QLC combines immediate reward and future reward to calculate the integrated Q-value. Firstly, it calculates the immediate reward accord to the topology and key information of pages; then, it draws future reward from learning experience; finally, it obtains Q-value from the combination of immediate and future reward.

In order to calculate reasonable integrated Q-value in QLC, the first is to solve the problem of how to weight the dependent degree of immediate and future reward. So, we introduce the weight function.

Definition: \( S \) is the set of states, \( A \) is the set of action, \( R \) is the reward space, \( W \) is the weight space. IRF \( (s, a) \) is the immediate reward function of taking action \( a \in A \) in the state \( s \in S \), and FRF \( (s, a) \) is the future reward function. The mapping from \( R \) to \( B \) could be expressed WF: \( R^R \rightarrow W \), we call it weight function. It is shown as follow:

\[
W_f(s,a) = \alpha \times \text{IrA}(s,a) + \beta \times \text{FrA}(s,a)
\]

\( \alpha \) and \( \beta \) are reward factors, and \( 0 \leq \alpha, \beta \leq 1 \), \( \alpha + \beta = 1 \). The return value of WF is called weight value.

The weight value returned from \( W_f(s,a) \) reflects the integrated Q-value of action \( a \) in \( s \) state. Reward factor reflects the dependent degree of immediate reward and future reward. If \( \alpha > \beta \), it means that QLC dependent on immediate reward more; if \( \alpha < \beta \), it means that QLC pay more attention on future reward.

Base on the analysis above, we propose the algorithm of reinforcement learning as follow.
Algorithm of form crawler based on reinforcement learning

Initialize:
set reward factor: \( \alpha \cdot \lambda, \beta = 1 - \lambda \). \( \lambda \) is discount factor, \( 0.5 \leq \lambda < 1 \)
set the lowest relevance: \( R \)

Input:
queue of URL
for each URL in queue do
Page = Query( URL );
If FormExist( page ) == TURE
form = ExtractForm( page );
RecordForm(form);
\( \alpha = \lambda \times \alpha, \beta = 1 - \alpha \); //Adjust reward factor
end if
urlset = ExtractUrl( page ); //process page
for each url in urlset do
IR = Irf(page,url);
FR = Frf(pages,url,Q-value table);
WF = \( \alpha \times \text{IR} + \beta \times \text{FR} \);
If WF <= \( R \) then
\( \alpha = \lambda \times \alpha, \beta = 1 - \alpha \); //Adjust reward factor
else
AddUrltoQueue(url);
UpdateQvalueTable();
\( \alpha = \lambda \times \alpha, \beta = 1 - \alpha \); //Adjust reward factor
end if
end for
end for

Algorithm always selected the highest confidence degree links according to the weigh value. Using this crawling strategy could avoid crawler drop into the case of local optimum. In the early search period, we assign immediate reward factor \( \alpha \) a larger value, so the crawler could discover the target link set quickly, and then factor \( \alpha \) decreases gradually with factor \( \beta \) value increases, increasing the dependence of future reward.

3.3 Architecture of QLC

In the exploration phase, form crawler calculate integrated Q-value of each URL, then add the URL that meets the relevance requirement to the crawling task queue according to the Q-value. Crawler engine crawls and analyze the corresponding pages following the order of the task links queue, and extract form from page finally.

QLC's main logic flow is shown in Finger2.

Finger 2. QLC's Main Logic Flow

In the process of exploration, there are two options for QLC according to integrated Q-value of URL: If integrated Q-value satisfy the relevance requirements, the crawler update the Q-value table.
and add the URL into the of the library and relevant URL into task links queue to be crawled; on the contrary, it adds the URL to the standby queue.

3.4. Vulnerabilities Detection

In this section, we present the detection method for XSS (Cross-site scripting) and SQLI (SQL injection). Simulating web attacking and analyzing the data of response are used to detect these two kinds of vulnerabilities. Through the process of "EP simulating attacking test -> Feedback Analysis", the detection purpose will be achieved.

3.4.1. XSS Vulnerabilities Detection

XSS [3, 24] is typically found in web applications that enable malicious attackers to inject client-side script into web pages viewed by other users.

Using "EP simulating attacking test -> Feedback analysis" method to detect XSS vulnerabilities, we construct some testing scripts according to the characteristics of EP, and then submit the query binding with testing scripts to the web application, wait for the program's response. Through analyzing the response for the query, the existence of XSS vulnerabilities could be determined.

Currently, some mechanisms have been used to prevent XSS vulnerabilities occur, such as data filtering or encrypting. However, these mechanisms still can’t put an end to XSS vulnerabilities [14, 19].

In our detection system, in order to ensure comprehensive detection, we have gathered a large number of various types of XSS vulnerability testing scripts. The following Table1 shows some examples of testing scripts.

<table>
<thead>
<tr>
<th></th>
<th>Examples Of XSS Testing Scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>&lt;SCRIPT&gt;alert(1)&lt;/SCRIPT&gt;</code></td>
</tr>
<tr>
<td>2</td>
<td><code>&quot;;:=&quot;&lt;SCRIPT&gt;alert(1)&quot;; JavaScript:alert(XSS)&gt;</code></td>
</tr>
<tr>
<td>3</td>
<td><code>&lt;IMG SRC=JaVaScRiPt:alert('XSS')&gt;</code></td>
</tr>
<tr>
<td>4</td>
<td><code>&lt;BODY ONLOAD=alert('XSS')&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>...</code></td>
</tr>
</tbody>
</table>

3.4.2. SQLI Vulnerabilities Detection

SQLI vulnerability is present when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and thereby unexpectedly executed [7, 23]. Hackers use malicious values insert into web forms or URLs to construct SQL command through a SQLI.

We also use "EP simulating attacking test -> Feedback analysis" approach for detecting SQLI. The detail of the approach has been proposed in [13], which is one of my other papers.

4. System Implementation & Experiment Results

4.1. System Implementation

To study the effects of the mechanisms proposed above, we implemented a vulnerability detection system in c#. The architecture of the detection system is showed in Finger3. The system is divided into three main functional models which include Q-learning crawler [QLC], vulnerability detector and database. The QLC takes the function of interaction between detection system and web application, it has the ability of generating web request, getting web response and analyzing web pages.. The vulnerability detector is responsible for the attacking code constructing, data of response analyzing and vulnerability identifying finally. Database is used for containing many kinds of information, including authenticated data and detection rules.
4.2. Experiment Results

In order to evaluate detection method proposed in this paper, we make the following definitions:

- **Crawling Coverage**: the ratio of the number of the crawled form and the total number of form in web application.
- **Crawling Yield**: the ratio of the number of the crawled form and the number of the crawled page.
- **Detection Coverage**: the ratio the number of and the total number of vulnerability existing in web application.
- **Detection Precision**: the ratio of the number of verified vulnerability and number of reported vulnerability.

4.2.1. Performance Analyzing Of Form Crawling

While testing the performance of form crawling, we gathered a set of metadata that include 100 URLs, all of which point to page containing form. According to the form type, the metadata was divided into four categories: search query class, login authentication class, information submitted classes and other types class, as the same as present in 3.2.1. After training of QLC, we chose 10 web sites as testing objects, and compared the performance of form crawling between traditional index crawler [22] and QLC. The experimental results are shown in Finger 4 and Finger 5.
Experimental results show that: at the situation of no access limitation for detection objects, the coverage of index crawler is little more than QLC, but the crawling yield of index crawler is so lower. After training effectively, QLC gains a much higher crawling yield than index crawler, which is nearly 200%~400% times higher. As this result, the efficiency of detection will be increased more.

4.2.2. Performance Analyzing Of Vulnerability Detection

Based on the crawling results above, we detected the security vulnerabilities focusing on XSS and SQLI. In order to analyze the detection precision, we verified the detection results manually. The detection results are present in Table 4.

<table>
<thead>
<tr>
<th>XSS</th>
<th>SQLI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report by detection system</td>
<td>Verified manually</td>
</tr>
<tr>
<td>S1</td>
<td>5</td>
</tr>
<tr>
<td>S2</td>
<td>7</td>
</tr>
<tr>
<td>S3</td>
<td>1</td>
</tr>
<tr>
<td>S4</td>
<td>5</td>
</tr>
<tr>
<td>S5</td>
<td>4</td>
</tr>
<tr>
<td>S6</td>
<td>0</td>
</tr>
<tr>
<td>S7</td>
<td>6</td>
</tr>
<tr>
<td>S8</td>
<td>4</td>
</tr>
<tr>
<td>S9</td>
<td>7</td>
</tr>
<tr>
<td>S10</td>
<td>3</td>
</tr>
</tbody>
</table>

To test the detection coverage, we established a testing web site, in which we constructed a number of vulnerabilities accord to OWASP[3] and verified these vulnerabilities manually. And then we used the detection system basing on QLC detect the testing web site. The table below shows the experimental results.

<table>
<thead>
<tr>
<th>XSS</th>
<th>SQLI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of form</td>
<td>96</td>
</tr>
<tr>
<td>Actual number of vul</td>
<td>21</td>
</tr>
<tr>
<td>Reported number of vul</td>
<td>17</td>
</tr>
<tr>
<td>Verified number of vul</td>
<td>17</td>
</tr>
<tr>
<td>Detection coverage</td>
<td>81%</td>
</tr>
</tbody>
</table>

The results show that: most of vulnerabilities could be discovered by our detection system, the detection coverage and the detection precision have reached more ideal level.

5. Conclusion and Future Works

A vulnerability detection method based on reinforcement learning is proposed in this paper. The target of studying the web vulnerability detection mechanisms is to enhance the ability of web scanner
and increase the detection efficiency and precision. Base on these mechanisms, we have implement vulnerability detection system, and the experiments shows well expected result.

In the future, our research will includes improving the effect of learning engine, analyzing the complex-form, constructing the attacking codes and analyzing the response.

6. References