Research on Optimization of Web Vulnerability Automatic Detection Technology Based on Semantic Recognition

Yonggang Li *, Min Han, Aiyi Cao, Shanmin Pan

State Grid Information and Communication Industry Group Co., Ltd, Beijing 102211, China.

*Corresponding author Email: Liyonggang08@163.com

Keywords: Web vulnerability, cross-site scripting, semantic recognition, automatic detection, optimization strategy.

Abstract: In recent years, with the rapid development of China's energy internet, the era of the Internet of Everything is coming. Coupled with the wide application of Web2.0, traditional client applications can no longer meet the increasing demand for network interconnection, and Web applications have gradually become mainstream. New web application security issues are becoming more and more prominent. Different web applications also have different vulnerabilities, which will be discovered by criminals and may be used maliciously. Especially in the era of energy internet, the fall of a system often threatens the overall energy security. In order to discover vulnerabilities in time and propose solutions, Web vulnerability automatic detection technology is also increasingly important. In response to the above problems, this paper proposes the optimization research of automatic vulnerability detection technology based on semantic recognition, applies the semantic recognition technology to the Web automatic detection technology and proposes a variety of optimization strategies, so as to obtain more comprehensive and accurate vulnerability information of Web applications. Improve the overall coverage and accuracy of web system vulnerability detection.

1. Introduction

With the in-depth development of energy Internet, three understandings of energy Internet have gradually formed in the world. From the perspectives of communication, software and cross-regional power grid, enterprises in various countries have formed their own unique direction of developing energy Internet. However, no matter from what angle, the Web application information systems of various specialties of energy Internet also developed rapidly actively or passively in the Web2.0 era. Due to various factors, such as technical reasons, man-made reasons, management reasons, etc., Web application systems have more or less security vulnerabilities. Cross-site scripting vulnerability (XSS vulnerability for short) is a serious Web vulnerability, which can cause serious harm to users and Webmasters in web applications with a large number of user interactions, including phishing, hijacking of user Cookie (identity verification marks of websites) and session hijacking. Web automatic detection technology aims at quickly and accurately detecting security problems in Web applications, helping enterprises to fix problems as soon as possible, and ensuring the information security of enterprises and users.

However, there are some shortcomings in the detection technology of XSS vulnerabilities in the traditional Web automatic detection technology. To solve this problem, this paper proposes a Web vulnerability automatic detection technology based on semantic recognition, and puts forward several optimization detection schemes. The specific experimental analysis proves that the detection efficiency and accuracy of this method are improved.

2. Cross site scripting vulnerability related introduction

Cross site script, also known as XSS vulnerability, is JavaScript code constructed by hackers. After being analyzed by Web information system, it is successfully executed in the client browser of

  DOI: 10.25236/isaiee.2020.014

normal users, which directly leads to the theft of sensitive data and information of users, thus achieving the purpose of attack.

Common XSS vulnerabilities can be classified into reflective XSS, storage XSS and DOM XSS. At present, there is no good solution for the automatic detection method of storage XSS. The main reason is that the automatic detection of storage XSS vulnerabilities will inevitably lead to the insertion of dirty data into the database, which will have a great impact on the business. The research direction of this paper is mainly aimed at reflection type and DOM type.

2.1 Reflection XSS vulnerability

Reflective XSS, also known as reflective cross-site scripting vulnerability, is a one-click and one-execution vulnerability. It can only be triggered once in a specific scene, and the generated location is the user's browser page. Generally, hackers need to construct a carefully designed URL link or HTML page, which will contain malicious code constructed by hackers. When the victim clicks the URL link or requests the HTML page, the malicious code is successfully returned to the browser after being parsed by the Web application and executed successfully.

The reflective XSS attack mode is shown in Figure 1:

1. User A logs into website B.
2. The B website has a reflective XSS vulnerability.
3. Attacker C detects the loopholes in website B and researches the attack vector.
4. Attacker C sends the URL link containing the attack vector to A through email phishing.
5. User A logs into website B and visits the URL link sent by attacker C.
6. The attack vector contained in the URL is executed on A's client browser. Through the execution of the attack vector, attacker C obtains user A's Cookie. With the unexpired Cookie, C successfully forged A's identity to obtain A's account permissions.

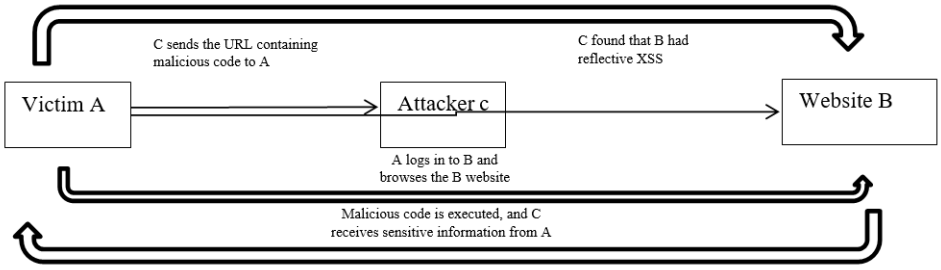

Figure 1 Reflection XSS Attack

The characteristic of reflective XSS is that the user's input and output are not filtered effectively, which can make the attacker's malicious code return to the client after being processed by the web server. The whole process is visually similar to a reflective behavior, so it is called reflective XSS.

2.2 DOM XSS vulnerability

From the effect point of view, DOM XSS is the same as reflective XSS, which is triggered once in a specific scene. However, the biggest difference between them is that the DOM-type XSS malicious code is completed before the server responds, and the browser's DOM parsing can directly trigger XSS. This type of vulnerability is mainly caused by unsafe calling of client script. Because DOM XSS relies on client DOM parsing, it is necessary to judge whether the attack vector exists in JavaScript code and detect whether there is syntax error.

The DOM-type XSS attack mode is shown in Figure 2:

1. Attacker C detects DOM-type XSS on website B and studies the attack vector.
2. C sends the URL link of the constructed attack vector to the victim A by means of email phishing.
3. A click on the link.

The attack vector was successfully executed in the browser of the attacker A. When A is unknown, attacker C quietly obtains A's Cookie. With the unexpired Cookie, C successfully obtains A's account permissions.

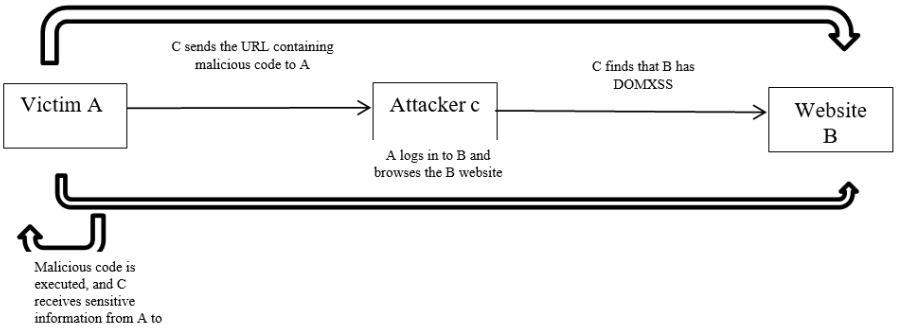

Figure 2 DOM XSS attack method

3. Method for detecting cross-site script vulnerability

Common full-automatic vulnerability detection tools include APPSCAN, AWVS, Burp Suite and other software, as well as many open-source tools dedicated to detecting XSS vulnerabilities on the Internet. Through the analysis of these tools, it can be concluded that the existing detection techniques include detection methods based on test vectors and detection methods based on regular matching. Traditional scanners basically use these two detection methods to automatically detect the existence of cross-site scripts.

3.1 Detection method based on test vector

Some fully automatic vulnerability detection tools are developed based on test vectors. The core steps of detection can be simplified as:

1. Select the URL to be detected from the queue.
2. Traverse the existing test vector and implement the attack.
3. Analyze the server response data packet and determine whether the XSS vulnerability exists according to whether the returned page contains a test vector.

It can be seen from the above steps that no matter whether the tested URL has a loophole or a waf, the test vector will be traversed to attack, which greatly consumes the detection cost and the detection efficiency is extremely low. In other words, the resources consumed by detection are directly proportional to the size of the test vector library, while the success rate is inversely proportional to the size of the test vector library.

3.2 Detection method based on regular matching

The method based on regular matching is used to detect DOM XSS vulnerabilities. The formation of DOM XSS vulnerability is caused by the nonstandard writing of client JavaScript code, so automatic detection can be realized by regularly matching sensitive functions in browser JavaScript files. For example, the common functions that can cause DOM XSS vulnerabilities are: eval, evaluate, execComand, assign, navigate, etc. The core detection steps are:

1. Select the URL to be tested from the queue.
2. Traverse the existing test vector and implement the attack.
3. Perform regular matching on the content of the JavaScript file in the user's browser. Determine whether there are XSS vulnerabilities based on whether the attack vector exists in a sensitive function that can execute the attack vector.

The accuracy of this detection method is relatively low. Because most functions are actually no problem, only in specific semantics can DOM-type XSS be caused, which leads to a lot of automatic detection results but a low hit rate.

3.3 Detection method based on semantic recognition

In order to improve detection efficiency, this paper introduces an automatic detection technology based on semantic recognition.

Semantic recognition here is based on python's native library -HTMLParser. The function of this library is to syntax the corresponding response package, so as to achieve the effect of quickly locating vulnerabilities. The core steps of detection are shown in Figure 3:

1. Send strings randomly.
2. Get the server response package, and analyze whether the matching contains the string.
3. If it does not exist, there is no loophole. If so, proceed to step 4.
4. Confirm the location of echo according to the syntax analysis of the response package.
5. According to different echo situations, different test vector libraries designed according to each situation are selected for detection.
6. Parse the response packet returned by the server to judge whether there are test tags or test attributes or JavaScript statements in the test vector library, and then judge whether XSS vulnerabilities exist.

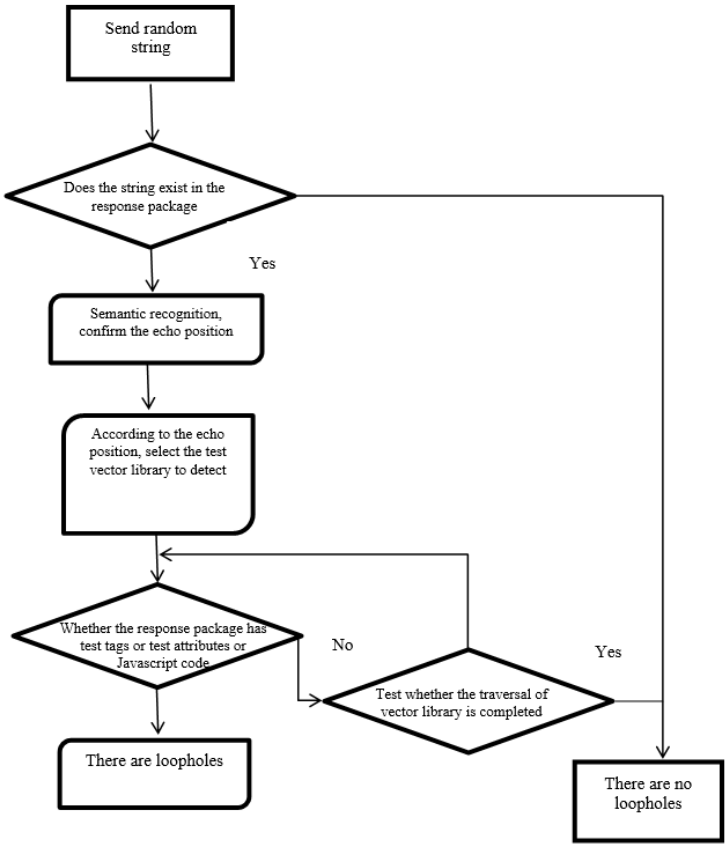

Figure 3 Detection method based on semantic recognition

Different from the traditional detection method relying on test vectors, semantic recognition can skillfully reduce the number of test vector libraries and greatly improve the accuracy of test vector libraries.

4. Optimization scheme of cross site script detection model

In order to optimize the detection effect of automatic detection, this paper combines the above three methods and proposes three other optimization strategies to effectively improve the detection accuracy and efficiency.

4.1 Built in hidden parameter optimization

The first step of the existing web vulnerability automatic detection software is to get the URL to be tested by crawler. The quality of the crawler directly affects whether the test result is comprehensive. If the URL with vulnerability is not crawled by the crawler, the vulnerability cannot be detected. Because there are many human factors in the development of WEB applications, it is inevitable that there will be test statements or uncommitted code segments in the source code. The functions of these code segments are invisible to users, which means that the crawler can't get the relevant URL, but these code segments may also have XSS vulnerabilities.

By collecting 100 GET type parameters that are common in web applications, the data set adopts the black cloud history vulnerability database, and extracts URL links and all URL parameters by traversing all articles, which are sorted according to the number of occurrences, and the top 100 with the most occurrences are taken as the fixed hidden parameter list in the model. Each probe traverses the hidden list to detect whether there are hidden parameters that can be echoed to the response packet. Take the top ten parameters with the highest frequency as an example, as shown in Figure 4:

id	6845
action	1643
type	1503
m	1013
a	992
c	855
act	829
page	813
uid	616
url	585
method	545

Figure 4 Detection method based on semantic recognition

At the same time, it will be found in the actual penetration test that there may also be hidden parameters in the FORM form in the response packet of the WEB server. In each probe, the parameter values in the FORM form are regularly matched and compared with the built-in hidden parameter list. If it is not in the list, the probe request is added.

4.2 Annotation optimization

It is not enough to judge whether there is grammatical structure. If the sent test vector appears in the annotated code, it may not be triggered because the attack vector is annotated, which needs to be divided into the following two situations:

Reflected in comments of JavaScript code

```
<html>
 <body>
 <script>
 var rage = 1;
 // inline <?php echo $_GET["rage "];?>
 </script>
 </body>
 </html>
```

According to semantic recognition, the output position of rage parameter is located in JavaScript structure. If it is not dealt with, the model will produce false positives. Because the finally inserted test vector will be annotated, it cannot be executed.

Therefore, it is necessary to use '\n' for detection first. If the backend does not filter '\n', it means that the annotator is bypassed. The test vector can be successfully executed in the next line, otherwise there is no loophole.

Reflected in the comments of HTML code

```
<!--
  <?php echo $type;?>
-->
```

According to semantic recognition, the output location of the type parameter is in the HTML structure. If left untreated, the model will produce false positives.

Need to use "-->" or "--!>" for detection first. If the backend does not filter, the comment symbol is bypassed. The test vector can be successfully executed in the statement below the comment, otherwise there is no loophole.

4.3 Test vector optimization

Traditional fixed test vector library has few changes and low efficiency, and one of the biggest hidden dangers is that it cannot bypass WAF. Some WAFs adopt the blacklist method, and if the HTTP request is detected to carry a specific test vector, the WAF will return a specific page. Once WEB applications are equipped with WAF products, the efficiency of traditional scanners based on test vectors will be very low. Some WAF will record the number of attacks, and ip will be blocked when the threshold is reached.

In order to solve the problem of WAF, it is necessary to solve the problem that aggressive test vectors are detected by WAF. This article uses the traditional test vector method of random string replacement to optimize. Take "</textarea>*" as an example below, * means test vector output bit.

The detection steps are as follows:

1. send random case character string to the server, judge whether the character string exists in the response packet, and then judge whether the test vector is echoed.

2. Judging the grammatical structure of echo position according to semantic recognition.

3. Judging whether the tag should be closed according to the grammatical structure.

4. Judging whether quotation marks are needed according to grammatical structure

5. According to the grammatical structure, judge whether annotation closure is needed.

6. Take the closed tag "</textarea>" as an example. First generate a random case string, structure such as: "</textarea>" + "<random case string1> generated random size string2"

7. According to the semantics of the response packet, identify whether a string tag of random size is newly added to the syntax structure. If it exists, the tag can be inserted. The next step is to detect whether the keyword is filtered.

8. The structure is such as: "</textarea>" + "<Generate random case string1> Generated random size string2", where the random size string contains key attributes or functions executed in the test statement that may be intercepted by WAF. For example: "iNpUTFoCuSoNfOcUsOnclickasdfaKasdW", "SCRiptSRc=jAvAsCrIpTasdfasOPOF" etc. This includes attributes or tags that may be included in test vectors such as input, onclick, onfocus, and script.

9.If step 8 is not intercepted by WAF, you can further detect more realistic test vectors. For example, "</textarea><img src=x>", judge whether the execution is successful according to the syntax structure of the returned request.

According to the detection result of each step, the detection result of the URL is output, which is divided into three levels: no loopholes; loopholes may exist, and WAF may exist; loopholes and no WAF exist.

5. Experimental design and evaluation

5.1 Experimental data set

In order to prove the optimization effect, the experiment needs to be prepared to attack the shooting range. However, the open source attack range currently does not have a range specifically for cross-site scripting. This article selected 6 domestic open source php content management website source codes to build the shooting range, respectively: yzmcms, semcms, php168, zzcms, cmsms and qbcms.

In order to improve the test effect, the release version of the shooting range is not the latest version, but the version with loopholes in history.

The hardware and software configuration environment of this experiment is shown in Table 1.

Table 1 Experimental environment configuration

Experimental environment	Configuration
Processor	Intel Core i7-8750H
Frequency	2.2GHz
Memory	16GB
Operating system	Windows 10
Software environment	Python, awvs

5.2 Experimental design

In order to prove that the semantic recognition model proposed in this paper is superior to the traditional vulnerability detection model, it must be compared with the traditional vulnerability scanner. This paper chooses the traditional vulnerability scanner -AWVS for comparison.

The detection model proposed in this paper does not involve the crawler framework for obtaining the URL to be detected, so the active and passive crawler framework in the traditional vulnerability scanner is still selected. However, the crawlers of different vulnerability scanners are different, and the URL queues actively obtained by different crawler frameworks are also different. Based on such a situation, this paper adopts the following scheme:

1.AWVS uses active detection to detect vulnerabilities. And the proxy mode is started, and the detected traffic passes through the detection model proposed in this paper through the proxy.

2. The detection model proposed in this paper accepts AWVS proxy traffic, generates detection URL queue, and conducts vulnerability detection.

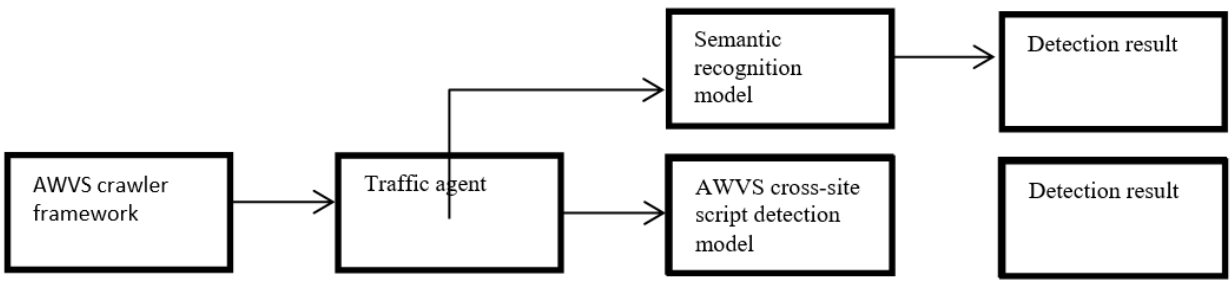

Figure 5 Experimental scheme

In the experiment, both detection methods have passed through the same URL queue. The final vulnerability detection situation can reflect the advantages and disadvantages of the model more accurately.

5.3 Model evaluation results

After detecting the vulnerabilities of six different open source php content management systems, the statistics of detection results are shown in Table 2.

Table 2 Automatic detection results of cross-site script

Content management system	Number of AWVS vulnerabilities detected	The number of vulnerabilities detected in this paper	AWVS vulnerability detection time/s	Vulnerability detection time of this paper's detection model/s
yzmcms	2	3	648	501
semcms	2	2	1102	283
php168	16	27	1700	445
zzcms	0	4	2234	804
cmsms	0	0	446	145
qbcms	6	10	7200	600

Experimental data show that the traffic detected by AWVS can really improve the detection efficiency and accuracy after passing through the detection model in this paper. The total number of cross-site script vulnerabilities detected by AWVS is 26. The number of detection models in this paper is 46, the number of increase is 20, and the promotion percentage is 76.92%. The detection time is also greatly reduced, and the average detection time of AWVS is 1226s (excluding abnormal samples). The average detection time of this detection model is 463s, and the average shortening time percentage is 62.23%.

6. Conclusions

In this paper, the XSS vulnerabilities of Web programs are introduced, and the shortcomings of traditional XSS vulnerabilities automatic detection are elaborated in detail. A vulnerability detection model based on semantic recognition is proposed. At the same time, three optimization schemes of detection models are proposed, and finally an experimental method based on agent strategy is designed. The experimental results show that the detection accuracy of this detection model is greatly improved and the detection time is effectively shortened compared with the traditional vulnerability detection model in detecting cross-site script vulnerabilities. In the follow-up work, it is still necessary to optimize the detection model, including crawler framework and automatic detection of stored XSS vulnerabilities.

Acknowledgments

This work is supported by State Grid Corporation Headquarters Science and Technology Project Funding (Research on Intelligent Detection and Verification Technology of Hidden Troubles in Electric Power Information Network, No. SGTJDK00DWJS1900105)

References

[1] Zhou Kangcheng. Research and Design of Web Application Vulnerability Scanning Tool[J]. Intelligent Computers and Applications, 2019, 9(04): 177-179.

[2] Zhang Huilin, Li Guancheng, Ding Yu, Duan Lei, Han Xinhui, Xiao Jianguo. A delimiter-based defense method for cross-site scripting attacks[J]. Journal of Peking University (Natural Science Edition), 2018, 54(02): 320 -330.

[3] Feng Kai. Web client attack detection and defense research [D]. Hunan University, 2011.

[4] Cao Yu. Research and implementation of XSS vulnerability detection tools based on fuzzing and web crawlers [D]. Beijing University of Posts and Telecommunications, 2018.

[5] Chen Jianqing, Zhang Yuqing. Design and implementation of Web cross-site scripting vulnerability detection tool[J]. Computer Engineering, 2010, 36(06): 152-154+157

[6] Wu Yaobin, Wang Ke, Long Yuehong. Network vulnerability attack and prevention based on cross-site scripting[J]. Computer System Applications, 2008(01):38-40+44.

[7] Pan Gubing, Zhou Yanhui. Discovery of XSS vulnerabilities based on static analysis and dynamic detection [J]. Computer Science, 2012, 39(S1): 51-53+85.

[8] Wei Cuntang. Research on the key technologies of SQL injection and XSS attack automatic detection [D]. Beijing University of Posts and Telecommunications, 2015.

[9] Feng Yitong. Research and design of XSS vulnerability detection system based on attack vector automatic generation [D]. Beijing University of Posts and Telecommunications, 2019.

[10] Li Yang, Zhou Jingshu, Yang Qing. Exploration of XSS attack mechanism and defense technology[J]. China New Communications, 2018, 20(19): 45.

[11] Ma Futian. Research and Implementation of Cross-Site Scripting Vulnerability Detection Technology for Web Applications [D]. Jiangnan University, 2018.

[12] Zan Jiawei, Yang Yong. Research on defense technology of cross-site scripting attacks based on DOM tree [J]. Communication and Information Technology, 2018(03): 62-67.

[13] Tang Qiyi. Research and implementation of fuzzing technology for rich text cross-site scripting vulnerabilities[D]. Beijing University of Posts and Telecommunications, 2018.

[14] Gao Yan, Bao Yongwu. Design and implementation of DOM-based cross-site scripting attack defense [J]. Network Security Technology and Application, 2018 (01): 18-19.

[15] Pan Jinkun. Research on Cross-Site Scripting Vulnerability Detection Technology [D]. National University of Defense Technology, 2017.

[16] Huang Nana, Wan Liang, Deng Xunkun, Yi Huifan. A cross-site scripting vulnerability detection technology based on sequence minimum optimization algorithm[J]. Information Network Security, 2017(10): 55-62.

[17] Mo Yonghua, Yu Bingbing. The design of XSS attack detection system in QR code[J]. Modern Computer (Professional Edition), 2016(24): 70-74.

[18] Gu Mingchang. Research on Cross-Site Scripting Vulnerability Detection Method Based on Penetration Testing [D]. Beijing University of Technology, 2016.

[19] Liu Da. Research on Defending XSS Cross-Site Scripting Attacks Through HTML Coding[J]. Cyberspace Security, 2016, 7(06): 23-24+31.

[20] Liu Jinhui, Ge Lina, Zhang Jing, Zhao Kai. Research on XSS Vulnerability Mining Technology Based on Fuzzy Testing [J]. Network New Media Technology, 2016, 5(01): 11-18.