# Review of XSS Attack and Detection on Web Applications

Wen-bing ZHAO [1,*], Dan WANG[1] and Zhi-ming DING[1]

[1]College of Computer Science, Beijing University of Technology, Beijing 100124, China

*Corresponding author e-mail: zhaowb@bjut.edu.cn

**Keywords:** XSS Attack, Vulnerability Detection, Web Application Style.

**Abstract.** Aiming at the difficulties to prevent Web applications to be maliciously injected which are increased by all kinds of dynamic Web technologies applied, concentrate on XSS attack, this paper reviews the research progresses of Web application injection vulnerabilities detection in recent years. It summarizes the classification and causes of the XSS injection security vulnerabilities, analyzes the complexity of security vulnerabilities detection; then proposes the key technologies of the existing detection approached, including analyzing and identifying the injection points, injection detection by software analysis and testing, symbolic execution, taint analysis; finally presents its future development direction.

## Introduction

With the rapid development of Internet technology, Web applications are widely used in all aspects of life. Unfortunately, the Web security technology is not so strong that Web applications are often maliciously attacked. It is the complexity of the Web application structure and dynamic Web technology[3], that greatly increases the complexity to detect the Web application injection and proposes the challenge to the research of the problem [4-6].

First, users interacting with Web applications in more flexible and dynamic ways[7-8] also provides attackers many more hidden approaches for illegal operation and attack. Second, browser script sets up many obstacles for attack detection. Third, the dynamic characteristic of JavaScript language and Ajax technology fulfill users' real-time, interactive and responsive demand. However, they make the architecture of Web applications more complicated to detect injection vulnerabilities.

According to OWASP [1], considering different attack sources, injection attacks of Web application include SQL injection, XSS (Cross Site Scripting) attack, XML injection and XPATH injection. Among these, XSS attackers use website loopholes to embed malicious HTML or client Java Script codes into Web pages. When users browse the Web pages, the malicious codes will be executed to steal user's Cookies, hijack user's sessions, phishing or pharming[10]. XSS attacks nearly become the most serious threaten to the safety of Web applications. This article reviews the classifications, causes and detection of XSS attacks.

## The Classification of XSS Attack

### Reflected Type

Attackers forge links containing malicious codes and entice users into opening the links. Once the link is opened, the malicious codes are executed. Because it can only be triggered by click, it is called reflected. Such XSS attacks are often hidden in the places such as web search bars, log entries, etc. to steal client Cookies or phish.

#### Stored Type

It is also called persistent type. Attackers upload and store malicious codes in the server, hiding in the comments, message boards, log or other interaction interfaces. As long as users browse the page containing malicious codes, it is executed without user's click on the particular link.

#### DOM-based Type

Document Object Model (DOM) is a W3C recommendation of dealing with the extensible markup language standard programming interface, independent with browsers, platforms and languages. DOM is often used as the output of Java Script page. The malicious codes of this type are hidden in nodes of the DOM document. They need not to be parsed and executed by the server. When the browser parses the DOM document, the malicious codes in it are executed. This type of XSS attack occurs in the browser totally.

#### Flash Type

Many Web applications reference dynamic contents in the form of Flash. Flash files can perform Action Script which provides interfaces to communicate with Java Script. So dynamic Flash files is often used to execute malicious codes to realize XSS attacks.

## The Causes of XSS Attack

### Same-origin Policy

Same-origin policy demands dynamic contents can only be read or modified within the homologous the HTTP replies or cookies. This policy is used by all browsers supporting JavaScript. But in order to reduce the inconvenience brought by homologous restriction, it allows different sub-domains of the same field to visit each other between pages. Using this kind of violation, once attackers injected malicious Java Script codes to one page successfully, it can influence all of other pages of the same field.

Moreover, the same-origin policy allows cross-domain applications of Java Script Scripts or image elements. This weakness may be used by attackers to pass parameters in a GET sentence, for sending the privacy information to the specified target link and returning the corresponding Java Script codes depending on the user's type. In this way, cross-domain-two-way communications are achieved by XSS attacks.

Even HTML 5, released in October 2014, is inevitable attacked by XSS [34].

### Cookie Security Strategy

The cookie security strategy is also based on the same-origin policy, so the security properties of Cookie can be completely bypassed through Java Script Script. Since cookies contain users' privacy information, they are always the ultimate goals for most XSS attacks. Cookies' security attribute HttpOnly sets up the transmission only at the HTTP level, the client unable to use the Java Script function *document.Cookie* to speaking, read and write the cookie. This prevents XSS attacks partly. However, the server's response may still make XSS easily access the cookies which setting the HttpOnly attribute in the same domain.

In order to protect users' privacy information, W3C released P3P (platform for privacy preferences project) specification. But it may lead to a paradox situation: if websites use P3P policy effectively, the browser will allow a third party's Web requests automatically with cookies, which increases the risk of XSS attacks; if the site is not set P3P policy or the P3P policy is invalid, a third-party Web request will not be with cookies from the Web site, thus eliminate the possibility of XSS attacks.

### Attacks through Flash

Because of supporting multimedia, Flash becomes the more popular plug-in of browser than Java Script. Assisted by Flash, XSS attacks are not only more hidden but also can realize more functions then by Java Script. For example, create connections to all of the domains the original TCP Socket can be connected and forge the header information of any HTTP request, thus it can scan the intranet computers and ports which cannot be accessed from outside.

## Key Technologies of XSS Attack Detection

### Injection Point Analysis and Recognition

Web application's architecture, content of pages, events, links, and the structure characteristics, etc., such factors are all helpful for injection point analysis and recognition.

The traditional static crawler accesses the target site's directory structure and parses the source codes of each page to get a potential injection points list, then analyze the server's response data through the feature matching to determine which are XSS injection vulnerabilities[30]. Parsing Java Script codes on the client-side can crawl pages in Ajax applications[31]. The crawled XSS vulnerabilities can be automatically repaired[32].Analyzing the source codes of Web applications may find more injection points than the crawling method[12].

With Web pages become more and more complex, injection point may not only exists in the returned pages after Web requests, but also can be created by users' events. Meanwhile, inputs leading to Web applications' data flow and control flow variability, which make it difficult to accurately obtain injection points' information, especially for the hidden interfaces. Therefore, new mathematical methods such as fuzzy logic, threat modeling are introduced to evaluate potential risk of injection vulnerabilities [33, 35].

### Detection Method Based on Software Analysis

Static software analysis technology can find potential security risks in the developing phase. It has the high coverage rate on the source codes, but often the high rate of false positives and limited source types. The static tools such as Pixy [16], can carry on the effective detection of XSS attacks in the PHP source code, provide the assignment sequence and slice based on data dependence.

Dynamic one includes control and data flow analyses, observing the execution of the program, the memory usage and dynamic properties such as register values to find vulnerabilities. It will not produce false positives and is high precision, but its test coverage rate is low[11]. The current comprehensive analysis tools include Fortify [13], CodeSecure [14], the Rational Software Analyzer [15], etc.

### Detection Method Based on Software Testing

After discovering the potential injection points, it can be tested if the XSS attack vulnerability really exists by inputting the test data through the suspicious interface and observing the response of the Web application. This is the method of software testing to detect XSS attacks. The white box testing is used to track the flow of data, while the black box to collect the feedback from the server and judge the effect of the input.

The software testing for venerability detection is mostly focused on test information collection, response analysis, improving test adequacy and accuracy, and performance evaluation[12, 17]. Other related researches focus on the efficient generation of test data, recognition of test interface, and how to determine whether the actual output according with expectations[8, 18] . The popular black box testing tools about Web application security testing include IBM AppScan [29], HP WebInspect [20], OWASP open source project WebScarab [21], etc.

### Detection Method Based on Symbolic Execution

There are some symbolic execution tools based automatic generation of test data, which can be used to detect Web application vulnerabilities to consider the feasible paths adequately and generate more accurate information[9], such as EXE [22], KLEE [23], JPF-SE[24], S2E[25], etc.. Most of them are concentrated in the processing of traditional C program. The S2E can detect vulnerabilities of binary program.

The reflected and stored XSS attacks in the context of PHP and Java can be detected by inspecting the execution[17]. The lack of input validation on the client side, which may cause operation space explosion problem, can be automatically discovered in Java Script running space[7]. The HTML form input constraints on the client side can be found to generate inputs, which can be used in dynamic symbolic execution to validate the server side behaviors. The problem is difficult to guarantee the full coverage of the interface[19].

### Detection Method Based on Stain Analysis

The main idea using stain analysis to detect vulnerabilities is to mark untrusted sources of data as the tainted and tracking these data in order to find security vulnerabilities. TAJ [26] uses dynamic stain analysis to detect vulnerabilities in Web applications written in Java.

By piling in binary level to track information flow, the high accuracy can be obtained on the reconstruct of attacks and signature extraction[27]. Work[2] is a tool to detect the browser’s vulnerabilities by enhanced stain analysis, guided by a dynamic stain penetration testing to find the client's Java Script injection attack, depending on manual test cases from external. Work [28] is oriented XSS vulnerability detection, by dynamic analysis to track the sensitive information in Web browsers.

## Conclusions

As to preventing the XSS attack, it is important to establish a comprehensive and quickly response injection-points-found mechanism in Web application environment. This is an integral part of Web security.

The highly automatic vulnerability detection needs more intensive research. It includes automatic construction of detailed vulnerability model and rules to guide the vulnerability detection, automatic generation of test cases as input, automatic analysis of injection points, etc.

In the case of limited time and resources, integration of various technologies to ease the contradiction between the resource consumption and vulnerability analysis precision, is still the problems need to be solved.

## Acknowledgement

This research was financially supported by the National Science Foundation of China (No.91546111) and Beijing Municipal Education Commission (PXM2015_014204_500221).

## References

[1] Information on http://www.owasp.org.cn/owasp-project/download/OWASPTop102013V1.2.pdf.

[2] P. Saxena, S. Hanna, P. Poosankam, et al., FLAX: systematic discovery of client-side validation vulnerabilities in rich Web applications [C]//17th Annual Network and Distributed system security Symposium.San Diego:ISOC,2010:1-17.

[3] T. Scholte, W. Robertson, D. Balzarotti, et al., An empirical analysis of input validation mechanisms in Web applications and languages[C]//Proceedings of the 27th Annual ACM Symposium on Applied Computing.Trento :ACM,2012:1419-1426.

[4] W.G.J. Halfond, S. Anand, A Orso, Precise interface identification to improve testing and analysis of web applications[C]//Proceedings of the 18th International Symposium on Software Testing and Analysis. Chicago:ACM,2009:285-296.

[5] S. Thummalapenta, K.V. Lakshmi, S. Sinha, et al., Guided test generation for web applications [C]//Proceedings of the International Conference on Software Engineering.San Francisco:IEEE, 2013:162-171.

[6] M. Balduzzi, C.T. Gimenez, D. Balzarotti, et al., Automated discovery of parameter pollution vulnerabilities in Web applications [C]//Proceedings of Network and Distributed System Security Symposium. San Diego:ISOC,2011:1-10.

[7] S. Artzi, J. Dolby, S.H. Jensen, et al., A framework for automated testing of JavaScript Web applications [C]// Proceedings of International Conference on Software Engineering. Honolulu:IEEE,2011:571-580.

[8] A. Mesbah, A. van Deursen, S. Lenselink, Crawling Ajax-based Web applications through dynamic analysis of user interface state changes[J]. ACM Transactions on the Web,2012,6(1):3.

[9] G. Buehrer, B.W. Weide, P.A.G. Sivilotti, Using parse tree validation to prevent SQL injection attacks[C]// Proceedings of the 5th International Workshop on Software Engineering and Middleware. Lisbon:ACM, 2005:106-113.

[10] A. Ciampa, C.A. Visaggio, M.D. Penta, A heuristic-based approach for detecting SQL-injection vulnerabilities in Web applications[C]//Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems, Cape Town:IEEE,2010:43-49.

[11] M.S. Lam, M. Martin, B. Livshits, et al., Securing Web applications with static and dynamic information flow tracking[C]//Proceedings of the 2008 ACM SIGPLAN Symposium on Partial Evaluation and Semantics Based Program Manipulation. New York:ACM, 2008:3-12.

[12] W.G.J. Halfond, S.R. Choudhary, A. Orso, Improving penetration testing through static and dynamic analysis[C]// Proceedings of the Second IEEE International Conference on Software Testing, Verification and Validation. West Sussex: John Wiley and Sons Ltd, 2011:195-214.

[13] Information on http://www.itesting.cn/index.php?_m=mod_article&_a=article_

content&article_id=126.

[14] Information on http://www.mainway.net/chanpin/code_secure.html.

[15] Information on http://softadvice.informer.com/Rational_Appscan_Source_

Edition.html.

[16] N. Jovanovic, C. Kruegel, E. Kirda, Pixy: a static analysis tool for detecting Web application vulnerabilities[C]//Proceedings of 2006 IEEE Symposium on Security and Privacy. Oakland:IEEE, 2006:258-263.

[17] W.G.J. Halfond, S.R. Choudhary, A Orso, Penetration testing with improved input vector identification [C]//Proceedings of the 2nd International Conference on Software Testing, Verification and Validation. Piscataway:IEEE,2009: 346-355.

[18] S. Mcallister, E. Kirda, C. Kruegel, Leveraging user interactions for in-depth testing of Web applications [C]//Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection. Cambridge: Springer, 2008: 191-210.

[19] P. Bisht, T. Hinrichs, N. Skrupsky, et al., NoTamper: automatic blackbox detection of parameter tampering opportunities in Web applications[C]//Proceedings of the ACM Conference on Computer and Communications Security. Chicago:ACM, 2010:607-618.

[20] Information on http://www8.hp.com/cn/zh/software-solutions/asset/software-

asset-viewer.html?asset=936485&module=1830243&docname=4AA1-5363ENW.

[21] Information on https://www.owasp.org/index.php/ Category:OWASP_

WebScarab_Project.

[22] C. Cadar, V. Ganesh, P.M. Pawlowski, et al., EXE: automatically generating inputs of death[C]// ACM 13th Conference on Computer and Communication Security. New York:ACM,2006:322-335.

[23] C. Cadar, D. Dunbar, D. Engler, KLEE: unassisted and automatic generation of high-coverage tests for complex systems programs[C]// 8th USENIX Symposium on Operating Systems Design and Implementation. San Diego: USENIX association, 2008:209-224.

[24] S. Anand, C.S. Păsăreanu, W. Visser, JPF-SE: a symbolic execution extension to Java pathfinder[C]// Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems,Berlin:Springer-Verlag, 2007:134-138.

[25] V. Chipounov, V. Kuznetsov, G. Candea, S2E: a platform for in-vivo multi-path analysis of software systems[J]. ACM SIGARCH Computer Architecture News, 2011,39(1):265-278.

[26] Q. Huang, Q.K. Zeng, Taint propagation analysis and dynamic verification with information flow policy[J]. Journal of Software, 2011, 22(9):2036-2048.(in Chinese)

[27] J. Clause, A.Orso, Penumbra: automatically identifying failure-relevant inputs using dynamic tainting[C]//Proceedings of Symposium on Software Testing and Analysis. Chicago:ACM, 2009:19-23.

[28] A. Armando, R. Carbone, L. Compagna, et al., Model-checking driven security testing of Web-based applications[C]//Proceedings of the Third International Conference on Software Testing, Verification and Validation Workshops. Paris : IEEE, 2010:361-370.

[29] M. Martin, M.S. Lam, Automatic generation of XSS and SQL injection attacks with goal-directed model checking[C]// Proceedings of the 17th Conference on Security Symposium. San Jose:USENIX Association, 2008:31-43.

[30] J.M. Chen, C.L. Wu, An automated vulnerability scanner for injection attack based on injection point[C]//2010 International Computer Symposium. Tainan:IEEE, 2010:113-118.

[31] A.V. Deursen, A. Mesbah, A. Nederlof, et al. Crawl-based analysis of Web applications: Prospects and challenges[J]. Science of Computer Programming, 2015, 97:173-180.

[32] I. Parameshwaran, E. Budianto, S. Shinde, et al., Auto-patching DOM-based XSS at scale[C]//Proceedings of the 10th Joint Meeting on Foundations of Software Engineering. Bergamo:ACM,2015:272-283.

[33] H. Shahriar, H. Haddad, Risk assessment of code injection vulnerabilities using fuzzy logic-based system[C]//Proceedings of the 29th Annual ACM Symposium on Applied Computing. Gyeongju:ACM, 2014:1164-1170.

[34] X. Jin, X. Hu, K. Ying, et al, Code injection attacks on HTML5-based mobile apps: characterization, detection and mitigation[C]//Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. Scottsdale:ACM,2014:66-77.

[35] N. Kaur, P. Kaur, Mitigation of SQL injection attacks using threat modeling[J]. ACM SIGSOFT Software Engineering Notes, 2014, 39(6):1-6.