ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 5, May 2019

Review on Intrusion using Web Site Vulnerabilities with K-Nearest Neighbour

Neeraj Dagar [1], Pooja Yadav [2]

P.G. Student, Department of Computer Science & Engineering, SRCEM, Palwal, Haryana, India[1]

Assistant Professor, Department of Computer Science & Engineering, SRCEM, Palwal, Haryana, India [2]

ABSTRACT: Nowadays, the greatest danger to associations is security that originates from its open Web website and the Web-based applications found there. Dissimilar to interior just system administrations, for example, databases—which can be closed from the outside by means of firewalls—an open Web webpage is commonly available to any individual who needs to see it, making application security an issue. As systems have turned out to be increasingly secure, vulnerabilities in Web applications have definitely pulled in the consideration of programmers, both criminal and recreational, who have contrived procedures to misuse these openings. Truth be told, assaults upon the Web application layer currently surpass those led at the system level and can have results which are similarly as harming. Thus, under the plan, we proposed Exposing Security Measure for Web Site Vulnerabilities or Web Server Intrusion with Machine Learning Algorithm utilizing KNN to barge in and get imperative and private data from the site, information servers, web servers, and web application inclined to assaults and frail to ensure its bare essential to protect if vulnerabilities exists in your Web Server.

KEYWORDS: Machine Learning, Cross Site Scripts, SQL Injections, K-Nearest Neighbour, Web Server Intrusion.

I. INTRODUCTION

With cross script classification, each input has the potential to be associate attack vector, that doesn't occur with different vulnerability varieties. This leaves a lot of chance for one mistake to occur in an exceedingly program that otherwise protects the net application against Cross Server aspect Script Classification and Structured source language Injection additionally has several potential attack vectors. Despite the favoured opinion that Cross Server aspect Script Classification is well prevented, it's several subtleties and variants. Even solid applications will have flaws in them; contemplate non-standard browser behaviour that tries to 'fix the misshapen Script-lets, which could go by a filter that uses regular expressions. Finally, till early 2006, the PHP interpreter had a vulnerability within which it failed to quote error messages, however several researchers solely reported the surface-level resultant Cross Server aspect Script Classification rather than determining whether or not there was a unique primary vulnerability that crystal rectifier to the error the identical is portrayed below with description for prepared reference as under:-

1. Cross Server Side Script Classification flaws occur whenever an application takes user supplied data and sends it to a web browser without first validating or encoding that content. Cross Server Side Script Classification allows attackers to execute script in the victim's browser which can hijack user sessions, deface web sites, possibly introduce worms, etc.
2. Injection flaws, particularly SQL injection, are common in web applications. Injection occurs when user-supplied data is sent to an interpreter as part of a command or query. The attacker's hostile data tricks the interpreter into executing unintended commands or changing data.
3. Code vulnerable to remote file inclusion (RFI) allows attackers to include hostile code and data, resulting in devastating attacks, such as total server compromise. Malicious file execution attacks affect PHP, XML and any framework, which accepts filenames or files from users.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 5, May 2019

4. Applications can unintentionally leak information about their configuration, internal workings, or violate privacy through a variety of application problems. Attackers use this weakness to steal sensitive data, or conduct more serious attacks.
5. Data Leakage and Improper Error Handling Applications can incidentally spill information about their setup, internal exercises, or misuse insurance through a variety of usage issues. Aggressors use this weakness to take sensitive data, or direct progressively certified the intrusion.

II. LITERATURE REVIEW

Santa Clara, Calif [2] WhiteHat Security, the Web security company, today announced the 2015 edition of the WhiteHat Security Website Security Statistics Report, which provides a one-of-a-kind perspective on the state of website security and the issues that organizations must address in order to safely conduct business online. This years report found that while no true security best practices exist, the key is in identifying the security metrics that mean the most to the organization and focusing on those activities to remediate specific vulnerabilities.

Akhawe, B., Saxsena, F., & Weinberge, S. [5] depicts that the most research on XSS defense has focused on techniques for securing existing applications and re-architecting browser mechanisms, sanitization remains the industry-standard defense mechanism. By streamlining and automating XSS sanitization, web application frameworks stand in a good position to stop XSS but have received little research attention. In order to drive research on web frameworks, we systematically study the security of the XSS sanitization abstractions frameworks provide. We develop a novel model of the web browser and characterize the challenges of XSS sanitization. Based on the model, we systematically evaluate the XSS abstractions in 14 major commercially-used web frameworks. We find that frameworks often do not address critical parts of the XSS conundrum. We perform an empirical analysis of 8 large web applications to extract the requirements of sanitization primitives from the perspective of realworld applications. Our study shows that there is a wide gap between the abstractions provided by frameworks and the requirements of applications.

Balzarotti, D., Cova, M., Felmetsger, V., Jovanovic, N., Kirda, E., Kruegel, [6] depicted that, the web applications evolved in the last decades from simple scripts to multi-functional applications. Such complex web applications are prone to different types of security vulnerabilities that lead to data leakage or a compromise of the underlying web server. So called secondorder vulnerabilities occur when an attack payload is first stored by the application on the web server and then later on used in a security-critical operation. In this paper, we introduce the first automated static code analysis approach to detect second-order vulnerabilities and related multi-step exploits in web applications. By analyzing reads and writes to memory locations of the web server, we are able to identify unsanitized data flows by connecting input and output points of data in persistent data stores such as databases or session data. As a result, we identified 159 second-order vulnerabilities in six popular web applications such as the conference management systems HotCRP and OpenConf. Moreover, the analysis of web applications evaluated in related work revealed that we are able to detect several critical vulnerabilities previously missed.

Doupe, A., Cui, W., & Jakubowski, M. H. (2013) [7] depicted that, the web applications are constantly under attack. They are popular, typically accessible from anywhere on the Internet, and they can be abused as malware delivery systems. Cross-site scripting flaws are one of the most common types of vulnerabilities that are leveraged to compromise a web application and its users. A large set of cross-site scripting vulnerabilities originates from the browser's confusion between data and code. That is, untrusted data input to the web application is sent to the clients' browser, where it is then interpreted as code and executed. While new applications can be designed with code and data separated from the start, legacy web applications do not have that luxury. This paper presents a novel approach to securing legacy web applications by automatically and statically rewriting an application so that the code and data are clearly separated in its web pages. This transformation protects the application and its users from a large range of server-side cross-site scripting attacks. Moreover, the code and data separation can be efficiently enforced at run time via the Content Security Policy enforcement mechanism available in modern browsers. We implemented our approach in a tool, called deDacota, that operates on binary ASP.NET applications. We demonstrate on six real-world

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 5, May 2019

applications that our tool is able to automatically separate code and data, while keeping the application's semantics unchanged.

Duraisamy, A., Sathiyamoorthy, M., & Chandrasekar, S. (2013, March)[8] depicted that, the increasing dependence on web applications has made them a natural target for attackers. Among these attacks SQL Injection Attacks (SQLIA) and Cross-Site Scripting attacks are the most prevalent. Our SQL Injection detection method is based on the design of a detection tool for the HTTP request send by clients or users and look for attack signatures. The proposed filter is generic in the sense that it can be used with any web application. Finally we test our proposed security mechanism using the vulnerability scanner developed by us as well as other well-known scanners. Our approach for Cross-Site Scripting detection method describes the possibilities to filter JavaScript in Web applications in server side protection. Server side solution effectively protects against information leakage from the users environment. Cross-Site scripting attacks are easy to execute, but difficult to detect and prevent.

III. PROPOSED SCHEME

The proposed scheme starts by the task of code analysis where static analysis using parameter feature detection is performed first. The static analysis operates on python source code files where it analyses every file to determine specified vulnerabilities using payload vide KNN. The vulnerabilities are specified by the security rules which behave as the security knowledge for static analysis technique. Once the static analysis is completed, the next step is to perform the dynamic analysis on the web application using K Weight evaluated by KNN. The dynamic analysis carries out the testing process by the use of instrumentation technique. The instrumentation (payloads) approach is based on the idea that, the attacks occurring due to the input validation vulnerabilities can be handled by adding the validation to the source code by determining of instrumentation technique (payloads) of the original source code with the pre-defined instrumentation templates (payloads). Therefore, the instrumentation code would perform the validation on the input given at the runtime, as a result of which the attacks would be stopped from being carried out and also the attempt for an attack can be reported during the web applications runtime. To do this, an proposed scheme generates the instrumentation code based on the instrumentation templates that contains the specified templates for each target vulnerability type. Later on, the proposed scheme also inserts the generated instrumentation code into the original web application code automatically. For inserting the instrumentation code, the locations are extracted from the results produced by the static results. As, the instrumented source code, which is actually combination of the original source code and the instrumentation code, is executed the runtime attacks are formed as well as reported to the user. The architecture diagram of the proposed scheme is shown in Figure 1 The operations performed by the dynamic analysis agent for cross side scripts and sql injections are described in the points below:

1. The list of vulnerabilities is given as an input, along with the predefined instrumentation templates (payloads), to the instrument code generation agent.
2. The proposed scheme code generation agent generates appropriate instrumentation code (payloads) based on the vulnerabilities information provided by the vulnerabilities list. This information mainly includes the types of potential vulnerabilities, the location of vulnerabilities in the source code and the vulnerable method along with the vulnerable parameter of that method.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 5, May 2019

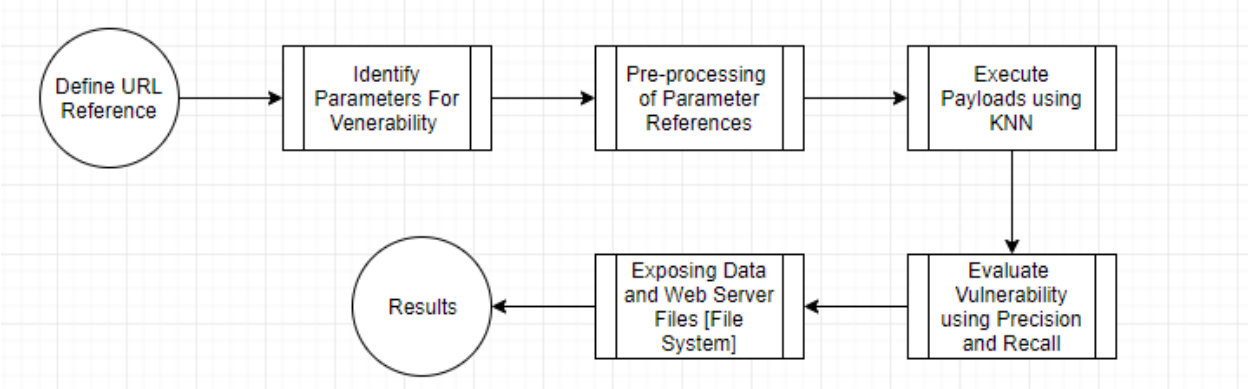

Figure 1: Work Flow of Proposed Scheme

IV. CONCLUSION

The web application worldview is as yet developing. Both client side and server side scripts are under dynamic advancement. Internet browsers as of late executed variety of scripts noteworthy adaptation of the language. New dialect components, for example, canvas, and expanded capacities, for example, cross-space and machine learning model like k-nn demands or relentless intrusion models, may concede the new abilities to perform penetration attacks on web server and data servers. Accordingly, the proposed XSS measures utilizing KNN must be persistently reexamined whether regardless they work given the present condition of the innovation. Likewise, the interruption limits may prompt the improvement of right now XSS Payloads and Sql-Injection assaults besides danger to the web world and with respect to white-hat the counter measures ought to be equipped.

REFERENCES

1. Alssir, F. T., & Ahmed, M. (2012). Web Security Testing Approaches: Comparison Framework. In Proceedings of the 2011 2nd International Congress on Computer Applications and Computational Science (pp. 163-169). Springer Berlin Heidelberg.
2. Antunes & Vieira (2012). Defending against web application vulnerabilities. Computer, (2), 66-72.
3. Bau, J., Bursztein, E., Gupta, D., & Mitchell, J. (2010). State of the art: Automated black-box web application vulnerability testing. In Security and Privacy (SP), 2010 IEEE Symposium on (pp. 332-345). IEEE.
4. Chen, S. (2014). wavsep. Available: http://sectooladdict.blogspot.com/2014/02/wavsep-web-application-scanner.html.
5. Dessiatnikoff, A., Akrout, R., Alata, E., Kaaniche, M., & Nicomette, V. (2011). A clustering approach for web vulnerabilities detection. InDependable Computing (PRDC), 2011 IEEE 17th Pacific Rim International Symposium on (pp. 194-203). IEEE.
6. Dougherty, C. (2012).Practical Identification of SQL Injection Vulnerabilities. 2012. US-CERT-United States Computer Emergency Readiness Team. Citado na, 34. . [Accessed: 08th June 2015]
7. Doupe, A., Cova, M., & Vigna, G. (2010). Why Johnny cant pentest: An analysis of black-box web vulnerability scanners. In Detection of Intrusions and Malware, and Vulnerability Assessment (pp. 111-131). Springer Berlin Heidelberg. [Accessed: 10th June 2015]
8. Fonseca, J., Vieira, M., & Madeira, H. (2014). Evaluation of Web Security Mechanisms using Vulnerability & Attack Injection. Dependable and Secure Computing, IEEE Transactions on, 11(5), 440-453.
9. Granville, K . (2015).Nine Recent Cyber-attacks against Big Businesses. New York Times [online] Available from http://www.nytimes.com/interactive/2015/02/05/technology/recent-cyberattacks.html?_r=1. [Accessed 08 July 2015.]
10. Howard, M., LeBlanc, D., & Viega, J. (2010). 24 deadly sins of software security [electronic book]: Programming flaws and how to fix them. New York: McGraw-Hill.
11. Jovanovic, N., Kruegel, C., & Pixy, E. K. (2010). A Static Analysis Tool for Detecting Web Application Vulnerabilities (Short Paper). In Proceedings of the 2006 IEEE symposium on Security and Privacy, Washington, DC, IEEE Computer Society (pp. 258-263).
12. Kalman., G. (2014). Ten Most Common Web Security Vulnerabilities.[online] Available from: http://www.toptal.com/security/10-most-common-web-security-vulnerabilities [Accessed 08 July 2015.]
13. Kals, S., Kirda, E., Kruegel, C., & Jovanovic, N. (2014). A web vulnerability scanner. In Proceedings of the 15th international conference on World Wide Web (pp. 247-256). ACM.
14. Khoury, N., Zavarsky, P., Lindskog, D., & Ruhl, R. (2011). Testing and assessing web vulnerability scanners for persistent SQL injection attacks.In Proceedings of the First International Workshop on Security and Privacy Preserving in e-Societies (pp. 12-18). ACM.
15. McQuade, K. (2014). Open Source Web Vulnerability Scanners: The Cost Effective Choice?. In Proceedings of the Conference for Information Systems Applied Research ISSN (Vol. 2167, p. 1508). [Accessed: 18th June 2015]