Nat. Volatiles & Essent. Oils, 2021; 8(4): 2799-2808

Design a Web Security Testing Mechanism by Using Semantic Comparision Method to Prevent Cross Site Scripting Attacks

[1] Shantanu Mukherjee, [2] Sandip Roy, [3] Pinaki Pratim Acharjya

[1,2] Brainware University, 398, Ramkrishnapur Road, Barasat, North 24 Parganas, Kolkata - 700 125, India .

[3] Haldia Institute of Technology, HIT College Rd, Kshudiram Nagar, Haldia, West Bengal, India.

Abstract

There are two faces of every coin, so is true for the web applications also. With the growing demand for online services web applications have become pervasive and with such exponential growth has increased the exploitation of the web application vulnerabilities also. JavaScript, which is the backbone in supporting a dynamic client side behavior, is often exploited for evil intentions. Malicious scripting code is injected into the application which are executed in the end user's computer thereby revealing the confidential parameters and compromising the security of the application. Cross Site Scripting (XSS) is certainly standing quite ahead in the hierarchy of the most dangerous and frequently used tactics executed by the attackers. Several defensive mechanisms are employed to prevent XSS attacks and reinforce the applications against them, however the attackers also are quite innovative and come up with newer attacking mechanisms. In this paper we try to apply Static Analysis to identify the XSS exploits. Although static application security testing mechanism comprehensively covers the entire code and is quite authentic in its vulnerability identification but, such good record is marred by a high number of false positives. Thus, we try to amalgamate static analysis and other algorithms to improve upon the cross-site scripting detection results.

Keywords: Static Security Testing, Cross Site Scripting, Web Application Security, Vulnerability Detection, Defense Mechanism, Client-Side Scripting, Security Attack

Introduction

The internet and the smart devices have been a harbinger of a silent revolution. Web applications are so widely used now-a-days that one cannot think of any other platform to transfer data and execute services through the web. Think of an industry - Banking and Financial Services, Technology, Hospitality, Education, Media or any other, the heavy reliance on the web technology is very evident. With such widespread use of the web technology, it has become a very attractive hunting ground for the hackers. The web applications house a whole universe of information and this is what the hackers try to access and manipulate to derive financial gains [1].

Cross-Site Scripting (XSS) is an attack, where a hacker is able to execute a malicious JavaScript code to exploit the vulnerabilities of an application. This can be used to, e.g., elicit confidential information like security parameters or manipulate content impersonating the victim [2][3]. Cross Site Scripting is one of the most vulnerable loopholes exploited by the hackers as per recent research [4] [5] [6].

Related Work

There has been a study on the threat perception of Persistent Client Side XSS caused by underestimating or overlooking the security of the storage at the client end. A taint analysis of the

malicious code execution flow from the cookies, which are the storage areas at the client end, to the sinks (JavaScript etc.) provides some insight into the problem [2]. So far cross site scripting is normally detected by studying the payload in a web request or a response, which is a single stage of a web transaction. However, J. Zhang et al. adopted a new approach of integrating evidences from a web request and its response to better classify XSS attacks [7]. Encoding the untrusted dynamic content is one of the best practices to prevent XSS vulnerabilities. In [8], the authors proposed a new approach that can fix common types of XSS exploits. In [9] the authors worked on summarizing the method of recognizing XSS based on the various machine learning algorithms and analysed their pros and cons. I.Tariq et al. have worked on Threat Intelligence to overcome XSS by Reinforcement Learning along with Genetic Algorithm [10]. Abikoye, O.C. et al. suggested curtailing the SQL injection and XSS attacks by devising a method for detecting and preventing these vulnerabilities using the Knuth-Morris-Pratt (KMP) string matching algorithm [11]. D. Zubarev et al. presented a summarised view of the problems associated with the cross-site scripting (XSS) in the graphical content of web-based applications[12]. The researchers in [13] presented a concept called WebMTD, a dynamic target protection technique that foils different types of XSS attacks on Web applications. M.Elkhodr et al. analysed the traditional methods used in preventing cross-site scripting and then proposed a security framework to improve the security of web applications against web-scripting attacks [14]. F. Caturano et al. designed an intelligent agent called Suggester that provides audit assistants to penetration testers so that the attack methods are sharper, more pointed and precise[15].

Cross Site Scripting

JavaScript is the main attack weapon in Cross Site Scripting attacks. The attacker injects JavaScript into the user's browser so that he is able to gain unauthorized access to sensitive information. Although XSS attacks are initiated at the client side, but the exploitation of the vulnerabilities occurs on the web server. The malicious script injected by the attacker is camouflaged as a harmless feature on the application's website thereby enabling the implementation of this harmful code within the website's trust field [16][17].

There was a common perception that Cross Site Scripting was of 3 types - Stored, Reflected, and DOM based, however in reality, there was a huge overlapping area that greyed out the clear distinction. Thus, starting about mid-2012, the research community proposed two broad classification of Cross Site Scripting - Server Side XSS and Client Side XSS.

Server Side XSS happens when a malicious data is sent by the user as a part of an HTTP response sent by the server. The origin of this data could be either from a stored location or from a request. This is called Server Side

XSS since the whole vulnerability is in the server-side code that sends the response to the browser and the browser innocently executes any valid script embedded in it.

Example of Server Side XSS:

Let's say a page outputs a variable directly from the database to an HTML page when generated:

<%=email %>

If email is not HTML encoded, it could be exploited by a malicious user by entering the email as

<script>sabotage Website();</script>

This will call the sabotage Website() function on page load thereby bringing down the website. Another type of Cross Site Scripting, the Client Side XSS is predominant when a malicious input updates the Document Object Model with an exploitable JavaScript function call. The loading of a web page results in the creation of a Document Object Model of the page. JavaScript can add, change, and remove HTML elements, HTML attributes, CSS styles, and HTML events. Any JavaScript call could be vulnerable if it can be leveraged to inject valid JavaScript into the DOM.

Figure 1: Flowchart of Client Side XSS

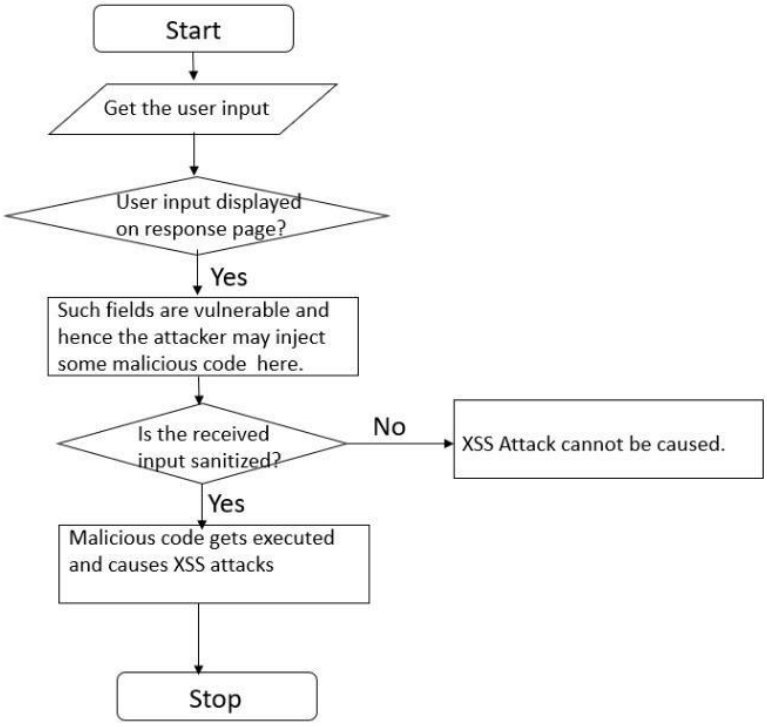
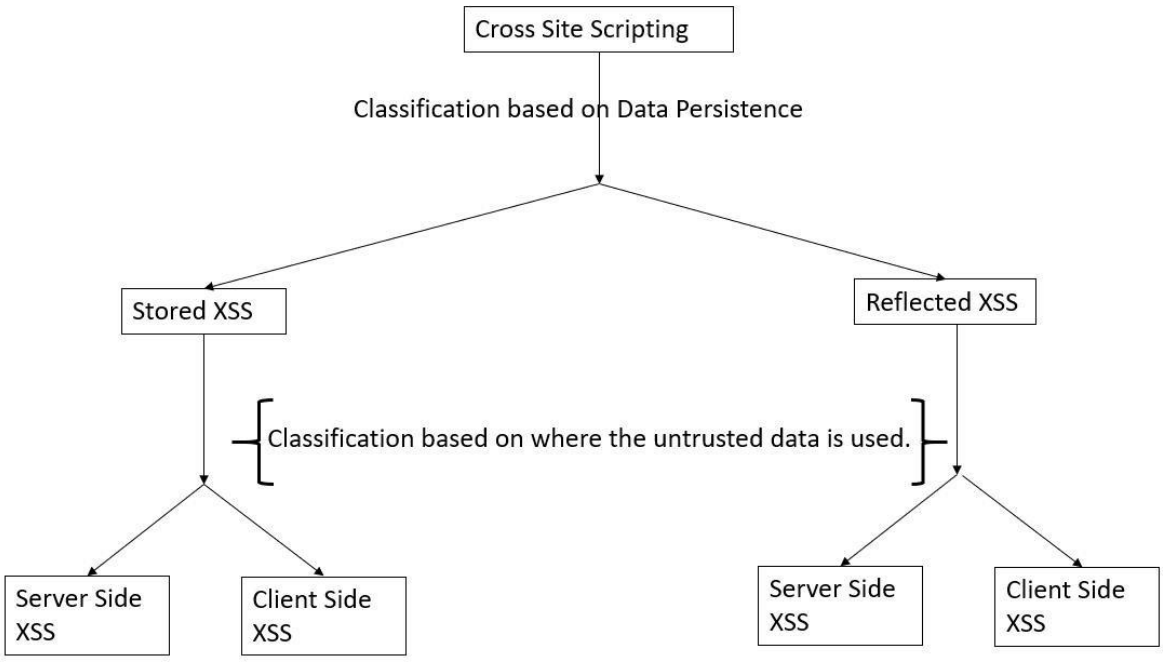


Figure 2. Types of XSS



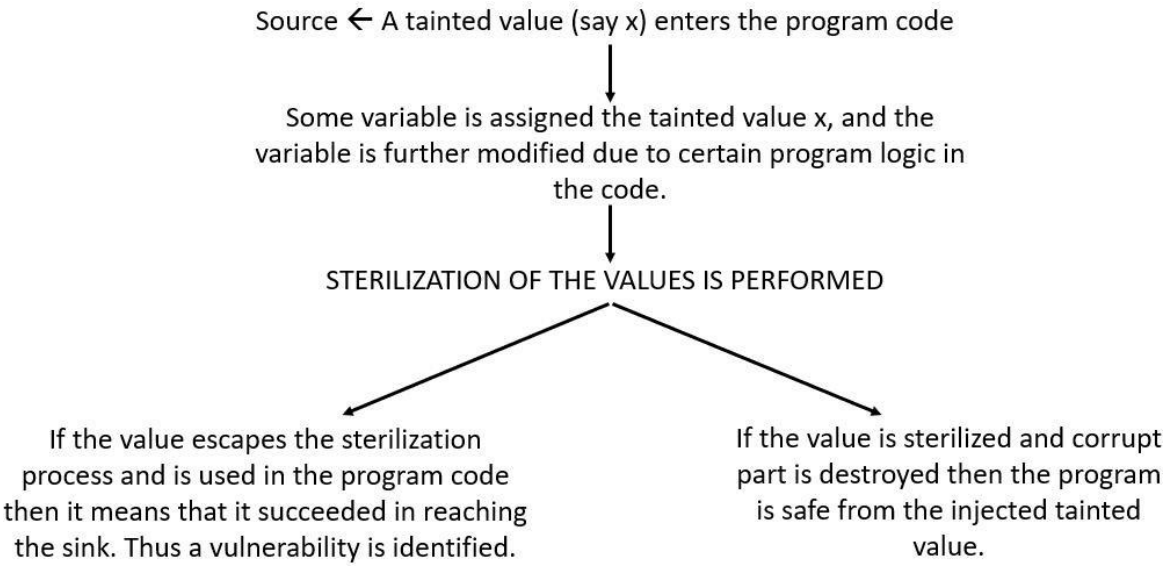
Xss Vulnerability Detection and Remediation

Model Specification and Parameter Estimation in Fixed Effect Model

Static Analysis: The static analysis approach analyses the complete source code of the application to scan for vulnerabilities. Although this approach is very reliable but it comes with a downside, that it identifies a high number of false positives [18]. Thus, once the vulnerabilities are identified the security testers and the developers have to slog again to confirm the findings of the static analysis approach and determine a remedial course of action [19].

Static analysis is somewhat analogous to a flow analysis problem, where it performs a dry run of the all the possible paths during the program execution. Thus, by injecting a corrupt value in this control flow graph and analysing its output can contribute in identifying the vulnerabilities.

Figure 3: Static Analysis to Detect XSS



Dynamic Analysis: Dynamic Analysis is more like black box testing. This is the process in which malicious inputs are bombarded to the web application and the resultant output is scrutinized for detecting the vulnerabilities [20] [21].

Dynamic analysis tools come up with a fewer number of false positives compared to the static tools, however these tools do not guarantee the detection of every single vulnerability since it may not execute whole program path thereby discovering vulnerabilities in only those paths which are covered in the course of the program execution [22] [23]

Amalgamation with Other Algorithms

Genetic Algorithm: Genetic Algorithm works on the concept of combining the best output to produce better results with every iteration and this iterative process stops when a stage is reached from where no further improvement in the result is possible [25] [26].

The GA procedure can be summed up as follows:

1. A Sample population is randomly generated. 2.For every (PV) in Potential_Vulnerabilities

a. While (PV) is not covered and the number of attempts is less than Maximum Number

i. Select a subset of population, most probable to produce the final solution

ii. Perform a crossover on the selected population to produce Offspring

iii. Generate a random population by mutation

iv. Increment the number of attempts

b. End While

End For

Chromosomes: We represent the individual input values for the application under test as chromosomes. The HTML requests for the web application under test make up the required individuals by encoding them in the respective URL.

Fitness Function: The fitness function measures the extent to which the chromosome is able to solve the issue under test [27]. Whenever an individual input goes into the application, it traverses various branches of the source code within that application. This measure of the number of branches traversed by the individual input is given by the fitness function. So, a higher fitness value is indicated if the input traverses 100% of the target branches in an exploitable path. So, if the input value covers all the branches of an exploitable path, then its coverage stands at 100% meaning a fitness value of 1, likewise, a 70% coverage of the exploitable path means a fitness value of 0.7. Hence, the fitness function is:

F(x) = ((Covered_Path_Percentage_% + Difference) * XSS_Percentage_%)/100
Covered_Path_Percentage_%: the proportion of the covered branches

Difference: the difference of the covered and the target path

XSS_Percentage_%: the percentage of the XSS patterns that is used to mark a test path by the Genetic Algorithm

Scenario Setup: We worked on the configurable parameters supported by Genetic Algorithms like the probability of applying mutation and crossover to individuals. Since they may impact the performance greatly, it became incumbent to spot the appropriate parameter first and its general impact on the performance. This paved the way for generating security test cases. We used APhpKb, PhpPlanner, Yapig and Mantis application for our case study.

Random Search Method: This is used more like a sanity check mechanism to gauge if the problem at hand is relatively simple and trivial approaches can be adopted to solve it. A randomly chosen set of input names are taken for the AUT to generate a random input value for each of those parameters.

Genetic Algorithm: We have taken a population of 80 individuals and studied their evolution over 600 generations. We filtered the 10% best individuals alive in each generation. Although we aimed to study 600 generations but we also applied an additional stopping condition whenever no improvement could be made after 200 consecutive generations, since it actually meant that a threshold of the search space has been reached by the algorithm from where no improvement could be possible.

Combined Strategy: Each of the above strategies had their own limitations and hence we worked upon combining these strategies to compensate for each other's limitations. So, logic applied was that of alternately applying a strategy until it reached its local optimum and then switching to the

other strategy and continuing with it till this other strategy reached its local optimum threshold and then again switching to the previous strategy. The stopping parameter remained the same, that is, either the test cases evolution happened for more than 600 generations or no improvement could be found for 200 generations consecutively.

Result Analysis

How fast is Genetic Algorithm when compared to other strategies in generating the security test cases? The table below shows the time taken by each of the approaches in generating the test cases for each of the application. For each of these approaches the table depicts the following parameters:

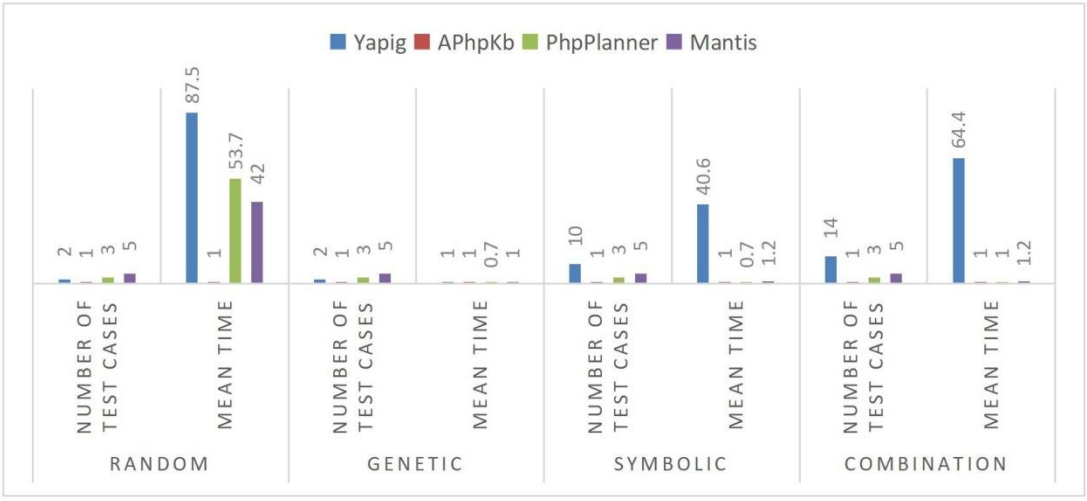
The number of test cases considered

The mean time (in seconds) to generate those test cases

Although the total number of test cases generated is quite high, but we have considered only one test case per vulnerability covered, for the purpose of analysis.

Tabular Data Analysis

APPLICATION	RANDOM		GENETIC		SYMBOLIC		COMBINATION	
	No. of Test Cases	Mean Time	No. of Test Cases	Mean Time	No. of Test Cases	Mean Time	No. of Test Cases	Mean Time
Yapig	2	87.5	2	1	10	40.6	14	64.4
APhpKb	1	1	1	1	1	1	1	1
PhpPlanner	3	53.7	3	0.7	3	0.7	3	1
Mantis	5	42	5	1	5	1.2	5	1.2



Conclusions

1. XSS vulnerability is by far one of the most exploited vulnerabilities applied by the hackers. This vulnerability leads to unauthorised access to user and site data and leading to extensive security violations and damages. Static code analysis is the most widely adopted technique to discover these vulnerabilities since such code analysis techniques aim to cover the entire code and the program flow.

2. GA scouts each viable path looking for the XSS vulnerability thereby contributing to improving the results of static analysis. However, this approach has not been found very suitable for web applications comprising of thousands of lines of code as the execution is manual in nature. If the program code is small, around 30- 40 lines of code then this method can be reliable.

3. This approach shows a glimmer of hope in reducing the high rate of false positives in static code analysis and encourages future researches to reduce it even further and strike an optimum balance.

Bibliography

Kaj U. Koskinen, PekkaPihlantob, HannuVanharantaa Pori.2003. Tacit knowledge acquisition and sharing in a project work context. International Journal of Project Management 21 (4): 281–290.

Oluwakemi C.A et al. A novel technique to prevent SQL injection and cross-site scripting attacks using Knuth-Morris-Pratt string match algorithm. EURASIP Journal on Information Security (2020) 2020:14, https://doi.org/10.1186/s13635-020-00113-y

Marius S. et al. Don't Trust The Locals: Investigating the Prevalence of Persistent Client-Side Cross-Site Scripting in the Wild. Network and Distributed Systems Security (NDSS) Symposium 2019, 24-27 February 2019, San Diego, CA, USA, ISBN 1-891562-55-X https://dx.doi.org/10.14722/ndss.2019.23009, www.ndss-symposium.org

Acunetix_web_application_vulnerability_report_2020. https://www.acunetix.com/white-papers/acunetix- web-application-vulnerability-report-2020/#cross-site-scripting-xss

Marashdih et al. (2018) Cross Site Scripting: Investigations in PHP Web Application. International Conference on Promising Electronic Technologies (ICPET)

https://www.veracode.com/security/ ; last accessed May 2021

https://owasp.org/www-project-top-ten/ ; last accessed May 2021

Zhang J., Jou Y.T. and Li X. Cross-Site Scripting (XSS) Detection Integrating Evidences in Multiple Stages. Proceedings of the 52nd Hawaii International Conference on System Sciences 2019

Mohammadi M., Chu B. and Lipford H.R. Automated Repair of Cross-Site Scripting Vulnerabilities through Unit Testing. IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Berlin, Germany, 2019, pp. 370-377, doi: 10.1109/ISSREW.2019.00098.

Chen X., Li M., Jiang Y., Sun Y. A Comparison of Machine Learning Algorithms for Detecting XSS Attacks.

(2019) Artificial Intelligence and Security. ICAIS 2019. Lecture Notes in Computer Science, vol 11635. Springer, Cham. https://doi.org/10.1007/978-3-030-24268-8_20

"Resolving cross-site scripting attacks through genetic algorithm and reinforcement learning, Expert Systems with Applications", I.Tariq, M.A.Sindhu, R.A.Abbasi, A.S.Khattak, O.Maqbool, G.F.Siddiqui, Volume 168, 2021, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2020.114386. (https://www.sciencedirect.com/science/article/pii/S0957417420310599)

Abikoye, O.C., Abubakar, A., Dokoro, A.H. et al. A novel technique to prevent SQL injection and cross-site scripting attacks using Knuth-Morris-Pratt string match algorithm. EURASIP Journal on Information Security 2020, 14 (2020). https://doi.org/10.1186/s13635-020-00113-y

Zubarev D. and Skarga-Bandurova I., Cross-Site Scripting for Graphic Data: Vulnerabilities and Prevention. (2019) 10th International Conference on Dependable Systems, Services and Technologies (DESSERT), Leeds, UK, 2019, pp. 154-160, doi: 10.1109/DESSERT.2019.8770043.

Niakanlahiji A. and Jafarian J.H. WebMTD: Defeating Cross-Site Scripting Attacks Using Moving Target Defense. Security and Communication Networks, vol. 2019, Article ID 2156906, 2019. https://doi.org/10.1155/2019/2156906

Elkhodr M., Patel J.K., Mahdavi M., Gide E. (2020) Prevention of Cross-Site Scripting Attacks in Web Applications. In: Barolli L., Amato F., Moscato F., Enokido T., Takizawa M. (eds) Web, Artificial Intelligence and Network Applications. WAINA 2020. Advances in Intelligent Systems and Computing, vol 1150. Springer, Cham. https://doi.org/10.1007/978-3-030-44038-1_100

Caturano F., Perrone G., Romano S.P., Discovering reflected cross-site scripting vulnerabilities using a multi-objective reinforcement learning environment. Computers & Security, Volume 103, 2021, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2021.102204.

Gupta S., and Gupta B.B. (2017) Cross-Site Scripting (XSS) Attacks and Defense Mechanisms: Classification and State-Of-The-Art. International Journal of System Assurance Engineering and Management, 8 (1): 512-530.

Algaith A. et al. Finding SQL injection and cross site scripting vulnerabilities with diverse static analysis tools. 14th European Dependable Computing Conference, IEEE Computer Society, Iasi, Romania, 10–14 September 2018.

Marashdih A.W., and Zaaba Z.F. Cross Site Scripting: Detection Approaches in Web Application. International Journal of Advanced Computer Science and Applications. 2016

Prokhorenko, Victor et al. Web Application Protection Techniques: A Taxonomy. (2016) Journal of Network and Computer Applications

Damodaran, Anusha et al. A Comparison of Static, Dynamic, and Hybrid Analysis for Malware Detection. (2017) Journal of Computer Virology and Hacking Techniques

Marashdih A.W. and Zaaba Z.F. Cross Site Scripting: Removing Approaches in Web Application. Procedia Computer Science, Volume 124, 2017, Pages 647-655, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2017.12.201.

Marashdih A.W., Zaaba Z.F. , and Omer H.K. Web Security: Detection of Cross Site Scripting in PHP Web Application Using Genetic Algorithm. (2017) International Journal of Advanced Computer Science and Applications

Nunes, P., Medeiros, I., Fonseca, J. et al. An empirical study on combining diverse static analysis tools for web security vulnerabilities based on development scenarios. Computing 101, 161–185

(2019). https://doi.org/10.1007/s00607-018-0664-z

Tariq I., Sindhu M.A. et al. Resolving cross-site scripting attacks through genetic algorithm and reinforcement learning. Expert Systems with Applications, Volume 168, 2021, 114386, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2020.114386.

Rodríguez G.E, Torres J.G et al. Cross-site scripting (XSS) attacks and mitigation: A survey. Computer Networks, Volume 166, 2020, 106960, ISSN 1389-1286. https://doi.org/10.1016/j.comnet.2019.106960.

Gan JM., Ling HY., Leau YB. (2021) A Review on Detection of Cross-Site Scripting Attacks (XSS) in Web Security. In: Anbar M., Abdullah N., Manickam S. (eds) Advances in Cyber Security. ACeS 2020. Communications in Computer and Information Science, vol 1347. Springer, Singapore. https://doi.org/10.1007/978-981-33-6835-4_45

Zhang, B.: Detecting XSS attacks by combining CNN with LSTM (2019). http://dx.doi.org/10.21227/css6- ds36

Alyasiri, H., Clark, J.A., Kudenko, D.: Evolutionary computation algorithms for detecting known and unknown attacks. In: Lanet, J.-L., Toma, C. (eds.) SECITC 2018. LNCS, vol. 11359, pp. 170–184. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-12942-2_14

Gupta R., Tanwar S. et al. Machine Learning Models for Secure Data Analytics: A taxonomy and threat model. Computer Communications, Volume 153, 2020, Pages 406-440, ISSN 0140-3664, https://doi.org/10.1016/j.comcom.2020.02.008.

Zhang J.M, Harman M. et al. Machine Learning Testing: Survey, Landscapes and Horizons. IEEE Transactions on Software Engineering, doi: 10.1109/TSE.2019.2962027Pathy, Kumar Chittarajan .D.R. (Ed). 2007. Forest, Government, and the Tribe. New Delhi: Concept of Publishing Company.

Ahmed, A. Kaleel, CB Senthilkumar, and S. Nallusamy. "Study on Amalgamation of Internet of Things in Industrial Applications." International Journal of Mechanical and Production Engineering Research and Development (IJMPERD) 8.1 (2018): 1279-1286.

Bleszynski, Jacek J., and Malgorzata Orlowska. "The Role of the Internet in the Free Time of Contemporary Poles." International Journal of Humanities and Social Sciences (IJHSS) 7.6 (2018): 69-76.

Mustafa, Faisal. "Collection Development in Libraries in Internet Era." International Journal of Library Science and Research (IJLSR) 5.2 (2015): 45-50.

Kavitha, A. K. "The Importance of Internet and Factors Influencing Internet Usage of Teaching Faculties-A Study with Special Reference to Selected Arts Colleges Affiliated to Bharathiar University." International Journal of Library Science and Research (IJLSR) 6.4 (2016) 11-16

Prashanthi, B., and S. Ratna Kumari. "Use of Mobile Phone and Internet: Adolescent Perceptions." International Journal of Communication and Media Studies (IJCMS) 7.4 (2017): 9-12.

Rathiga, K. "Attitude of Students towards Erudition of Vocabulary and Verbal Aptitude Using Computer Technology And Internet." International Journal of English and Literature (IJEL) 7.3 (2017) 55-62