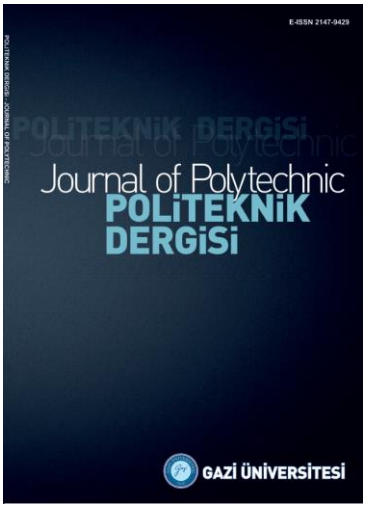


**POLİTEKNİK DERGİSİ**

***JOURNAL of POLYTECHNIC***

ISSN:1302-0900 (PRINT), ISSN: 2147-9429 (ONLINE)
URL: http://dergipark.gov.tr/politeknik


# Malicious xss code detection with decision tree

*Yazar(lar) (Author(s)): Ömer KASIM*

*ORCID: 0000-0003-4021-5412*

**Bu makaleye şu şekilde atıfta bulunabilirsiniz(To cite to this article)**: Kasım Ö., "Malicious XSS code detection with decision tree", ***Politeknik Dergisi***, 23(1): 67-72, (2020).

**Erişim linki(To link to this article):**http://dergipark.gov.tr/politeknik/archive

**DOI:10.2339/politeknik.470332**

# Malicious XSS Code Detection with Decision Tree

*Araştırma Makalesi / Research Article*

**Ömer KASIM***

Dumlupinar University, Simav Technology Faculty, Electric and Electronic Engineering, Kütahya, Turkey

(Geliş/Received : 14.10.2018 ; Kabul/Accepted : 20.02.2019)

**ABSTRACT**

Dynamic applications such as e-commerce, blogs, forums, e-governance, e-banking and portals that are in these platforms have become a part of our lives. However, a tremendous increase in the use of dynamic web and mobile applications has resulted in security vulnerabilities originating from the Hypertext Markup Language (HTML) coding system. Site-to-site Script Execution (XSS) attack is the largest contributors to security exploits. There are different models according to the dynamic content that XSS attacks use. The interest of the study is composed of attacks on visual content with the "img" tag. In study, an algorithm has been developed to detect XSS attacks with the decision tree which is motivated by the fact that they tend to be easier to implement and interpret than other quantitative data-driven methods. The algorithm that successfully classifies 392 of 400 malicious and clean codes in the data set with 8 different features. This result contributes to the use of secure internet without XSS attacks that use visual content..

**Keywords: Security vulnerability, XSS attacks, feature extraction, decision tree.**

## 1. INTRODUCTION

The dynamic transformation of web and mobile applications with Web 2.0 causes some vulnerability. These vulnerabilities are used to attack systems. These attacks are categorized by the Open Web Application Security Project (OWASP) [1]. In OWASP report, Cross-Site Script Execution attacks (XSS) are the most common attack type. It has been found that XSS attacks have also increased in parallel with an increase in the level of interaction of an application. The most important reason for this increase is due to the fact that the XSS commands operate at the application layer of the Open Systems Interconnection (OSI) architecture. This problem causes most existing intrusion detection systems to have difficulty detecting XSS attacks [1].

As shown in Figure 1, the attacker can use malicious JavaScript, VBScript, ActiveX, Hypertext Markup Language (HTML) or Flash code to attack the dynamic web page in the XSS process. Embedded HTML and JavaScript codes are the most used ones. In this process, it is possible to execute code to get access to data and cookies on the remote computer with XSS codes [2]. Information such as the user name and password of the user canbe obtained by accessing data and cookies on the remote computer.

XSS attacks are divided into three categories: Stored XSS Attack, Reflected XSS Attack and Document Object Model (DOM) [3]. The Stored XSS Attack is done by using fields such as the forum application and a text box that provides the visitor-book style interaction process in the web sites. XSS codes sent via these fields are recorded in the database. Thus, any user who uses the attacking application or visits the page is converted to XSS attack [4]. The Rejected XSS Attack is done with code entry into the Uniform Resource Locator (URL) fields. Users are searched for attacks by social media and URL addresses which is sent via an e-mail. The user who falls into this trap is in the target position by activating

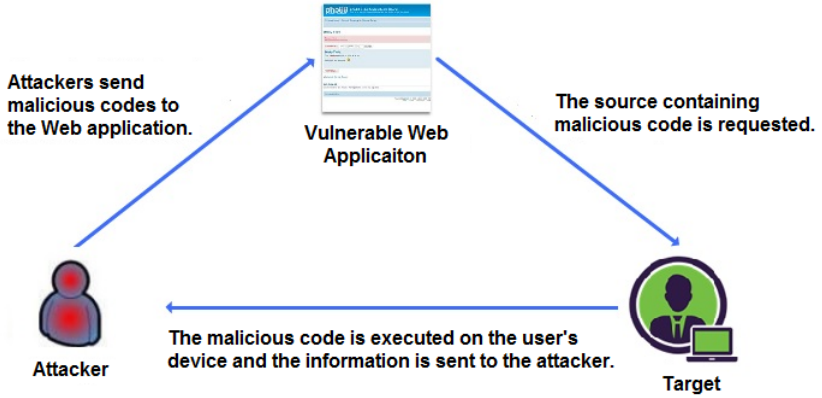


**Figure 1.** XSS Attack Scheme

the URL. The important point of this attack is that this attack can be seen only by the user who composed the XSS attack. The target user is in danger because of the

*Sorumlu Yazar (Corresponding Author)
e-posta : omer.kasim@dpu.edu.tr

unseen codes [5]. In the DOM XSS attack, the DOM field of the HTML page is changed. With the changing DOM, the payload field in Hyper Text Transfer Protocol (HTTP) is used for different purposes. The requested code in DOM is separated by the payload '#' symbol with the part in the URL.

This character is not received by the server. If malicious codes integrated after this character, this request is processed. The "document.referrer", which represents the URL of the uploading document, is directed with the DOM routing object. The payload is received by the server in the routing header information. Thus, the malicious codes are placed on the page by interfering with the code structure and index page of the web site. These operations are usually done with terminal-side JavaScript codes [6].

Risk reporting, using of SSL, login validation and authorization are used to protect the system from attacks during the design phase [7]. However, a malicious script running in the context of the current user, firewall, encryption method, or Intrusion Detection System (IDS) in XSS attacks may be ineffective in preventing these types of attacks [8]. Because, the XSS codes are run directly in the browser program that causing the application or web page to be damaged. Static, dynamic and hybrid approaches are used to solve this problem [9]. These solutions include some investment strategies in corporations and intuitions. These investments that can be applied in the process of making cyber security investment decisions are grouped into 5 categories. These categories are the determination and measurement of cyber security risks, measurement of the cost and effect of security attacks, measurement of the effectiveness of security technologies and determination of the optimal level of security investments [10]. To determine the investments, the malicious code analysis can be done.

Code propagation analysis, string test and software test tools are used in Static analyzes. One of them, code propagation analysis, is based on flow control structure [11]. Each node in the flow is represented by a label. When a decision variable of a node is observed, the web page is considered vulnerable or no problem in the running code. Another static analysis, String Analysis, uses the Context Free Grammar (CFG) structure [12]. CFG refers to the whole grammar or rules used in language design. Content in CFG structure consists of four features. These features consist of the ending ones, the continuous ones, the relations and the starting symbol. These 4 labels define the features of the string expressions. In the software-based test process, which is the third static analysis, error injection and penetration tests are used [13]. In dynamic analysis, XSS attacks are not tried to be detected with proxy based solutions. In this process, specific codes are allowed to operate according to the structure of the application and the service model [14]. In some cases, both static and dynamic structures are used together. This is the name given to the hybrid solution. Static and dynamic models are used together according to the application's service model and code structure [15].

There are different solutions in literature to overcome malicious codes which contains XSS. The easiest one is to disable the Javascript command execution process in browser programs. The other solution is Server-side programs. XSS-Guard is the one of them [16]. XSS Guard creates a shadow page for the actual response page of the programs on the server. This process makes it difficult for the attacker to kill or harm the process because the shadow page is accessed instead of the actual response page in the XSS attacks. On terminal side solution is proposed to prevent terminal-side High-Rate Distributed Denial of Service (HDDoS) attacks. In the first step of the work, the data is transformed by preprocessing, then the features of the captured data are extracted and in the last step the α-divergence test is performed. This process is applied to all incoming and outgoing packets through the proxy. When the distribution is examined, if the threshold value exceeds a predetermined value, the demand is not further processed and the process is terminated [17]. In the work of attack detection by following the proxy process, the parameter values of the request pages and the HTML and JavaScript codes of the response pages are analyzed. The method includes data collection, preprocessing, feature extraction and attack clustering on the most appropriate subset of related properties. In an attack, the JavaScript is disabled by the proposed program [18]. An artificial learning algorithm, on the other hand, detects XSS attacks with self-generated qualities on both static and dynamic structures. In proposed method, high false alarm rate and scaling problems have been observed in the detection process [19,20].

The proposed algorithm resolves the high false positive problem without having to disable Javascript and to make cyber security investment decisions. It is designed as server, client or proxy-side that provides .

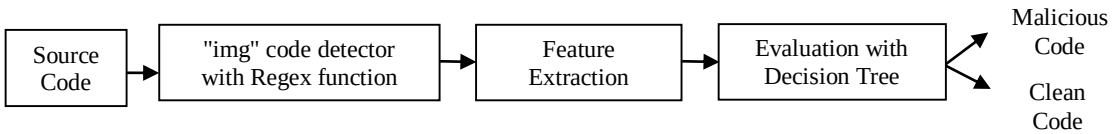


**Figure 2.** Block Diagram of Algorithm

- Initialize HTML request
- Initialize Regexp Function
- while the img tag token
- Calculate 8 Features
- while img tag matches
- If src,www,http,https are equals 1 and =/,#,hex are equals 0
- Print Code is clear, Keep Going Application or Web Page
- Else
- Print Code is not clear, Block the Web Page or Application

**Figure 3**. Pseudo Codes of the Proposed Algorithm

protection against dynamic and static XSS attacks by using the characteristics of the code. The decision tree method has been used as a classification technique and a predictive approach in order to analyses these characteristics. The trained decision tree algorithm classifies 400 sets of attributes with clean and maliciously code of visual html requests in the data set with 98% success. It supports secure internet usage by detecting XSS attacks to be done with html “img” tags.

## 2. PROPOSED APPROACHED

Block diagram of the proposed method which includes artificial learning algorithm with the decision tree structure is shown in Figure 2. The source code isreceived on request. Then the “img” label is detected by Regex function. If there is “img” label on request, it is analyzed by the proposed method. At analyze stage features of the code is obtained. Then the obtained features are given to the trained decision tree. At final stage the code status is determined as malicious or clean code. Also, the pseudo code of the algorithm is shown in Figure 3.

## 3. DETECTION OF XSS ATTACK

In the experimental part of the study, 3 steps are followed. These are the creation of the data set, feature extraction and evaluation steps.

### 3.1. Collection of Data

The malicious codes used in the training and testing phase of the study were taken from OWASP [21]. These malicious codes contain different XSS attack options.Our scope of in this study is about the malicious visual content. Therefore, the 350 malicious code line contains “img” labels. Then, our dataset is enriched with 50 clean code sets with clean "img" tags. The training and testing dataset of the decision tree is expressed on the Table 1.

**Table 1:** Training and Testing Dataset of Decision Tree

| | **Clean Code** | **Malicious Code** | **Codes in Dataset** |
|---|---|---|---|
| **Training** | 16 | 48 | 64 |
| **Testing** | 50 | 350 | 400 |

**Table 2:** Clean and Malicious Code Samples from Dataset

| | |
|---|---|
| **Clean Code** | <img id="resim" src=" http://www.xyz. org /images/ slide/slide-2/slide5.jpg "onmouseover= "hover(this);" onmouseout="unhover(this);" /> |
| **XSS Malicious Code Structure Disabling WAF** | <img src="x:x" onerror= "alert(XSS)"> |

The code samples of the “img” tag used in web operations that represent the visual content are in the data set of the study. Sample of clean and malicious code of these are shown in Table 2. The clean code which is in the Table 2 is effective in loading and using visual content. On the other hand, the "x: x" in XSS containing malicious code structure allows to exceed the firewall of the application.

### 3.2. Preprocessing and Feature Extraction

This step includes the creation of features to identify aggregated data. With the Regexp function, incoming and outgoing HTML requests are scanned on the terminal or server. For each request that is scanned, the tags areobtained in array form with the token and match sub-functions. When tags are defined with tokens, the parameters and values which are contained in the tags are obtained as a string array with the plug-in function. When “img” tags are derived from the token, the parameters and values of all the lines containing “img” tag are obtained with the matched plug-in. The code of the acquired visual content request contains the feature vector, these parameters and values.

**Table 3:** Features

| **Feature** | **Explanation of Feature** |
|---|---|
| $f_1$ | Img |
| $f_2$ | # |
| $f_3$ | Hex |
| $f_4$ | Src |
| $f_5$ | “=” After character “/” |
| $f_6$ | www. |
| $f_7$ | http:// |
| $f_8$ | https:// |

The 8 features obtained from the data set of the worker are described in Table 3. The $f_1$ feature changes to "1" if it is the "img" tag. When the "img" tag is not found, this feature information remains "0”. The second feature is the "#" character. This character is used before the functions in the distressed code lines are called. $f_2$ feature is "0" if it is not "1" in case of "#" character in “img”. In the case of $f_3$, these features are updated as "1", otherwise they remain as "0" in the feature vector. The $f_4$ measures the “src” characters. If these characters are composed of the together, this feature is set to “1”. Otherwise the $f_4$

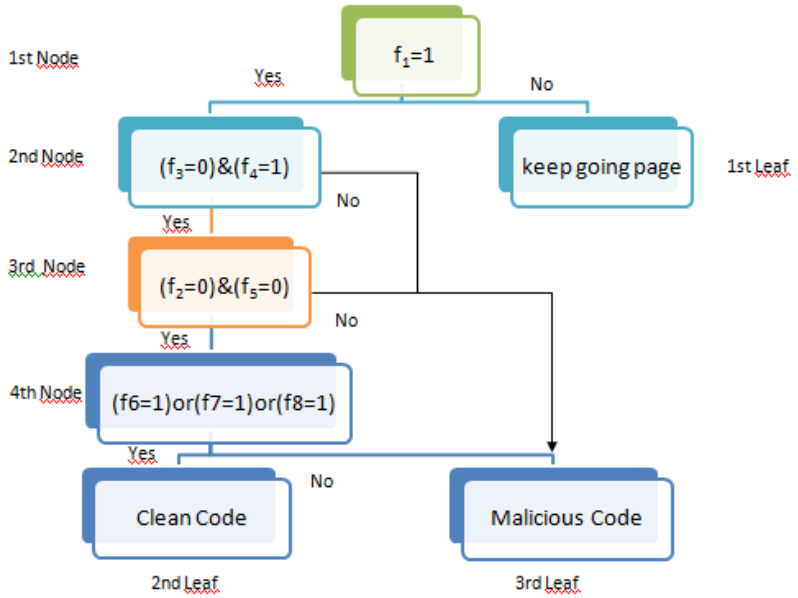


**Figure 4.** Structure of Decision Tree

feature is set to “0”. The = and / characters combination is the another XSS detect feature and it is represeted with $f_5$. If this combination is in the code then $f_5$ is set to 1. In the case of a visual request " $f_6$, $f_7$, and $f_8$ when the "src" part has "www", "http" or "https", the update is made in thefeature vector as "1" in case of these expressions and as "0" in case of these expressions.

### 3.3. Evaluation with Decision Tree

The decision tree algorithm tends to be easier to implement and interpret than other quantitative data-driven methods [22]. It starts with a single node that covers all the data in the training data set. The data in this node are divided into leafs and placed in the most appropriate class according to the division criterion. This criterion is to maximize the acquisition of information between random sub-samples of the dimensions obtained at each point. The splitting process continues until the minimum criterion or variance stop criterion is obtained. Decision trees and other classification algorithms have been tested and applied to malicious code detection. Generally, decision tree classifiers have been successfully created using data from honeypots and data from normal activities.This analysis leads to the creation of decision rules to identify malicious activity [23]. There are certain labels in the process of defining the XSS attacks in the OWASP report [1]. These labels are used to prepare the training vector of the decision tree. There are 64 raw data in training vector which is composed of 8 feature. The decision tree structure was formed with this vector.

When a decision tree of 8 features expressed in Table 3 is formed, a process as shown in Figure 4 is obtained from the dataset of the work. 8 feature information of each code line with the "img" tag is calculated and passed through the decision tree. Detection of malicious code is still being carried out at the request stage by obtaining output related to the tree branches.

Decision tree nodes are places where decisions are made in the flow. Leaves are the final step in which decisions are a result. There are 4 nodes and 3 leaves in the decision tree structure designed in the study. The "Code clean", "Code malicious" or "No visual content of the code" in terms of the decisions made at each node is expressed by Leaf 1, Leaf 2 and Leaf 3 respectively.

The first step in the decision tree is to look at the f1 feature in the first node. In case this value is "1", if 2nd node is "0", it goes to 1st leaf. At node 2, f3 and f4 are decided using feature information. If f3 is "0" and f4 is "1", the third node is going to 3rd leaf if neither or both of these conditions can be provided. If the values of the features f2 and f5 at node 3 are "0", the node 4 is directed to the leaf 3 in other cases. If the value of any one of f6, f7 and f8 at node 4 is "1", the analysis is terminated by going to leaf 3 if all values are "0" to leaf 2.

## 4. RESULTS AND DISCUSSION

XSS is one of the most used attack methods in the direction of OWASP's report [1]. These attacks, which are made through the means of interacting in the application, target the user's cookies. The ability of attackers to run malicious code on the application layer makes XSS detection difficult. XSS-style requests can be blocked by passing the request codes in the page or instantly through a filter. The proposed algorithm which is based on decision trees structure can detect these attacks that can be done with visual tags through “img” tags.

**Table 4:** Classification Result with Confusion Matrix

| | **Actual class** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Predicted Class** | **Features** | img | # | Hex | Src | "=" After "/" | www. | http:// | https:// | Clean Code |
| | img | 60 | | | | | | | | |
| | # | | 40 | | | | | | | |
| | Hex | | | 18 | 2 | | | | | |
| | Src | | | 2 | 48 | | | | | |
| | "=" After "/" | | | | | 30 | | | | |
| | www. | | | | | | 60 | | | |
| | http:// | | | | | | | 48 | 2 | |
| | https:// | | | | | | | | 38 | 2 |
| | Clean Code | | | | | | | | | 48 |

First, the code line with the "img" tag is determined by the Regexp function. With the analysis of the string expression in the "img" tag, it is decided by the tree leaf about the case that there is no hex expression after the trace of the word first. After analyzing "#" character and "=/" characters, "img" examines whether there is at least one of "www", "http: //" or "https: //" in triplet. If thedecisions in the process are positive for each node, the code is cleared. If any of this process is not successful, the code is malicious.

The data set containing 50 clean img code lines and 350 malicious img code lines has been successfully categorized by the algorithm developed to work cleanly or malicious when the code passed through the decision tree. The distribution of usage of the 400 data features in the data set of the study is shown in Figure 5. Classification Result with Confusion Matrix is obtained from the classification of this data set with decision tree is given in Table 4. 342 of the attack codes containing different tags in the data set were placed in the correct classes. On the other hand, the data containing 8 malicious code that have problems in classification are determined as http and https along with the attributes with "hex" and "src".

In the case of the detection of the compilation process to the hex codes of the malicious code and the absence of the src tag used next to the "img" tag, two lines of malicious code for the absence of "http" and "https" before the pages are specified as false negative and false positive. In order to determine clean code distribution with https, 2 lines of code are set to true negative. sensitivity is 98% and specificity is 96%. The results is shown at Table 5.

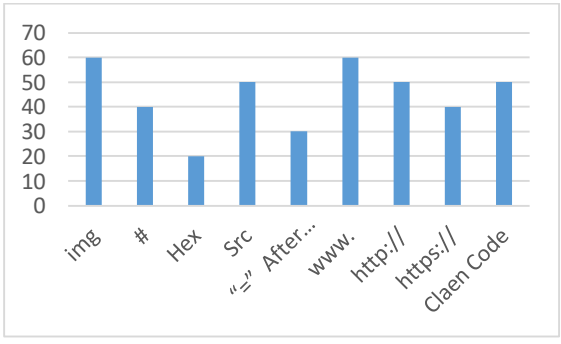


**Figure 5.** Frequency of Features in Dataset

To create a model, 50 data sets and 50 JavaScript keywords and symbols were created. XOS malicious code was detected using four classifiers, Naive Bayes [24], ADTree [25], SVM [26] and RIPPER [27]. In the test phase, classifiers were used to distinguish between normal and malicious code-containing attack commands. It is seen that the SVM classifier has the highest sensitivity and RIPPER has the highest recall value. In general, the average of malicious tagged scripts by a classifier is 90% [28].

**Table 5:** Table of Confusion Matrix

| | | **Actual class** | |
|---|---|---|---|
| | | **Malicious Code** | **Clean Code** |
| **Predicted class** | **Detected Code** | 342 True Positives | 48 False Positives |
| | **Non-Detected Code** | 6 False Negatives | 2 True Negatives |

The XSS attack system is implemented by using different fields in dynamic content. One of them is XSS attacks on visual content. "Filter bypass-based polyglot", "IMG OnError and JavaScript warning ciphers", and "WAF Bypass Strings for XSS" are XSS attacks using visual content [19]. In various studies, the proxies [13], request quantities of packages [15] and shadow pages instead of real pages [16] are examined. In an Attack Detection and Prevention system uses a whitelist, which may not be always enough can be detect stored XSS attacks [29]. However, proposed algorithm examines the codes at request code instantly. This process is provided protection by blocking the visual contents in case there is a negative attack by testing visual elements of request without loading. Each HTML request can be checked

instantly terminal and server sides or in a proxy. This is an important consequence of this work, where malicious code can be caught if malicious code is sent over the monitoring result visual content.

## REFERENCES


[1] https://www. owasp.org/ index.php/Top_10_2013-Top_10

[2] Garcia A., Navarro A., "Prevention of Cross-Site Scripting Attacks on Current Web Applications", ***Greece Proceedings of the OTM Confederated International***, 1770-1784, (2007).

[3] Imran Y., Pathan A., "Preventing Persistent Cross-Site Scripting (XSS) Attack by Applying Pattern Filtering Approach", ***IEEE The 5th International Conference on Information and Communication Technology***, 1-6, (2014).

[4] Jasmine M., Devi K., George G., "Detecting XSS Based Web Application Vulnerabilities", ***International Journal of Computer Technology & Applications***, 8(2): 291-297, (2017).

[5] Gupta, B., Gupta, S., Gangwar, S., Kumar, M., Meena, P., "Cross-Site Scripting (XSS) Abuse and Defense: Exploitation on Several Testing Bed Environments and Its Defense", ***Journal of Information Privacy and Security***, 11(2): 118-136, (2015).

[6] Dong R.Z., Ling J., Liu Y., "DOM Based XSS Detecting Method Based on Phantomjs", **Proceedings of the *International Conference on Applied Mechanics, Mechatronics and Intelligent Systems***, 237-241, (2015).

[7] Kasım Ö., "Evolving Web Process and Security", **9th *International Conference on Information Security and Cryptology***, 149-154, (2016).

[8] Yılmaz V., Sağıroğlu Ş., "Kurumsal Bilgi Güvenliği ve Standartları Üzerine Bir İnceleme", ***Gazi Üniversity Journal of Faculty of Engineering and Architecture***, 23(2):507-522, (2008).

[9] Saha S., "Consideration Points Detecting Cross-Site Scripting", ***International Journal of Computer Science and Information Security***, 4(1):1-8, (2009).

[10] Şentürk, H., Çil, C.Z., Sağiroğlu S., "Siber Güvenlik Yatırım Kararları Üzerine Literatür İncelemesi", ***Politeknik Dergisi***, 19(1): 39-51, (2016).

[11] Zou C.C., Gong W., Towsley D., "Code Red Worm Propagation Modeling and Analysis", ***Proceedings of the 9th ACM Conference on Computer and Communications Security***, 138-147, (2002).

[12] Prithvi B., Venkatakrishnan V.N., "XSS-GUARD: Precise Dynamic Prevention of Cross-Site Scripting Attacks", ***International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment***, 23-43, (2008).

[13] Baykara M., Daş R., Karadoğan İ, "Bilgi Güvenliği Sistemlerinde Kullanılan Araçların İncelenmesi", ***1st International Symposium on Digital Forensics and Security***, 27:231-239, (2013).

[14] Di Lucca G.A., Fasolino A.R., Mastoianni M., "Identifying Cross Site Scripting Vulnerabilities in Web Applications", ***Sixth IEEE International Workshop On Web Site Evolution***, 71-80, (2004).

[15] Bhuyan M.H., Bhattacharyya D.K., Kalita J.K., "Survey on Incremental Approaches for Network Anomaly Detection", ***International Journal of Communication Networks and Information Security (KUST),*** 3(3):226-239, (2011).

[16] Nithya, V., Pandian, S. L., Malarvizhi, C., "A Survey on Detection and Prevention of Cross-Site Scripting Attack", ***International Journal of Security and Its Applications***, 9(3): 139-152, (2015).

[17] Boro D., Bhattacharyya D.K., "Dyprosd: A Dynamic Protocol Specific Defense for High-Rate DDOS Flooding Attacks", ***Microsystem Technologies***, 23(3):593-611, (2017).

[18] Shahriar, H., Devendran V.K., Haddad H., "Proclick: A Framework for Testing Clickjacking Attacks in Web Applications", ***Proceedings of the 6th International Conference on Security of Information and Networks***, 144-151, (2013).

[19] Goswami S., Hoque N., Bhattacharyya D.K. "An Unsupervised Method for Detection of XSS Attack." ***International Journal of Network Security***, 19(5): 761-775, (2017).

[20] Likarish P., Jung E., Jo I., "Obfuscated Malicious Javascript Detection Using Classification Techniques", ***IEEE 4th International Conference on Malicious and Unwanted Software***, 47-54, (2009).

[21] https://www.owasp.org/index.php/XSS

[22] Abdallah I., Dertimanis V., Mylonas H., Tatsis K., Chatzi E., "Fault Diagnosis of Wind Turbine Structures Using Decision Tree Learning Algorithms with Big Data." ***Safety and Reliability–Safe Societies in a Changing World***, 3053-3061, (2018)

[23] Gregio A., Santos R., Montes A. "Evaluation of Data Mining Techniques for Suspicious Network Activity Classification Using Honeypots Data.", ***Proceedings of SPIR***, 6570: 1-10, (2007).

[24] John G.H., Langley P., "Estimating Continuous Distributions in Bayesian Classifiers", ***11th Conference on Uncertainty in Artificial Intelligence,*** 338–345, (1995).

[25] Freund Y., Mason L., "The Alternating Decision Tree Learning Algorithm", ***Proceeding of the 16th International Conference on Machine Learning***, 124–133, (1999).

[26] Platt J.C., "Using Analytic QP and Sparseness to Speed Training of Support Vector Machines," ***Advances in Neural Information Processing Systems***, 557-563, (1999).

[27] Cohen W., "Fast Effective Rule Induction", ***Proceedings of the 12th International Conference on Machine Learning***, 115–123, (1995).

[28] Sarmah U., Bhattacharyya D.K., Kalita J. K., "A survey of Detection Methods for XSS Attacks", ***Journal of Network and Computer Applications***, 118(15):113-143, (2018).

[29] Maurya S., "Positive Security Model based Server-side Solution for Prevention of Cross-site Scripting Attacks," ***Annual IEEE India Conference (INDICON),*** 1–5, (2015).