Cross-Site-Scripting Attacks and Their Prevention during Development

Ms. Daljit Kaur[1], Dr. Parminder Kaur[2]
Assistant Professor[1], Assistant Professor[2]
Department of Computer Science
Lyallpur Khalsa College,Jalandhar[1], Guru Nanak Dev University Amritsar, India[2]

Abstract— Web Applications are the important and popular software systems used in almost every era of life of the human beings. At the same time, there are large number of attacks on web applications that are getting popular among attackers. Attacks like injection vulnerabilities such as SQL Injection, Cross site Scripting, Cross site Request Forgery(CSRF) are very common and threatening to the modern web applications. This paper specially focuses on countermeasures of Cross Site Scripting (XSS) vulnerability. Here, we have implemented various attacks on a XSS vulnerable web application and also classified these countermeasures with respect to Software Development Life Cycle and tested them for their effectiveness with the help of vulnerability scanners. Finally, the result of vulnerability scanners are shown and analyzed before and after the implementation of known XSS countermeasures.

Keywords— Cross-Site-Scripting; XSS; Attacks; Vulnerability scanners; Threats; Web application; Security

I. INTRODUCTION

Web applications are popular on the web and so are the attacks on them. Hackers on the Internet have evolved from the greed of credit to deceit and stealing of the profitable data & identity.Security has been the critically important part of most of the web applications. the applications are often pierced with vulnerabilities, which are used by attackers to gain access to the assets of the web applications. Application layer has become the soft target of mostly attackers, as Gartner group research (http://www.gartner.com/technology/research.jsp) reported that 75% of hacks happen on web sites targeting the application layer than network, server, database, and web server layers. Securing the Web Applications is always a great challenge as they are at high risk [1]. Organizations are spending lot on the secure access to the web applications by using additional hardware and software but the real cause of insecurity of the web applications is the applications themselves. Symantec report reveals that 76% of the scanned websites are vulnerable and it also shows that in the year 2016, more than 1.1 billion identities were stolen in data breaches, almost double the number stolen in 2015, when just over 563 million identities were stolen. This is the despite the fact that the number of data breaches actually fell between 2015 and 2016—dropping from 1,211 to 1,209[2]. SQL injection (SQLi) and Cross-Site-Scripting (XSS) attacks continues the trend of previous years and have gained more popularity through the years[3,4]**.** OWASP (Open Web Application Security Project) has provided Top 10 List of the most critical risks of the web applications from time to time since 2003. There have been minor changes in the major risks of the web applications through the years. Threats like SQLi, XSS, Data Exposure and misconfigurations are still among the top[5,6,7]. Major web services such as Google Analytics, Facebook and Twitter have had XSS issues in recent years despite intense research on the subject [8,9].

This research paper focuses on XSS vulnerability and implements various attacks that can be performed on XSS vulnerable web applications and also implements existing XSS countermeasures in Software Development Life Cycle (SDLC) to check the effectiveness of them. Section II reviews the literature for known XSS Vulnerability, attacks and its countermeasures. Section III gives the brief overview of XSS vulnerability and its mitigation in development life cycle. Section IV implements various attacks in XSS vulnerable web application. In this research work, vulnerability scanner OWASP-ZAP (Zed Application Proxy) is used to confirm the presence of XSS vulnerability. This vulnerability scanners is available with operating system Kali Linux and also freely available at www.owasp.org. Kali Linux is an open source project that is maintained and funded by Offensive Security, a provider of world-class information security training and penetration testing services[10].Section V concludes the result and gives the future directions.

II. LITERATURE REVIEW

Juillerat, N. in [11], has shown that common vulnerabilities like SQL injection, XSS and URL injection in database web applications can be controlled by enforcing code security audit library in java . Finally he has compared his library-based approach to security with other available approaches and has shown that they don't only allow secure code to be written, but also enforce it.

Shalini S., et. al.in [12] have discussed XSS attacks and their prevention at client side. Mainly three categories of XSS: Reflected XSS, Stored XSS and DOM based XSS are considered in the proposed model's three step process. In new model, Script Detection, Analysing and Data Monitoring is done at client side in order to avoid XSS. Also the proposed solution is tested with many websites and browsers. The vast majority of XSS attacks can be prevented by identifying user input locations within the web application and ensuring the source code handling has proper measures in place.

Weinberger, J. ,et al.in [13] have studied the security provided in the XSS sanitization abstractions frameworks. A novel model of web browser and XSS sanitization challenges are also characterized. Based on the model, XSS abstractions in 14

major commercial web frameworks, are analysed and found that frameworks often do not address crucial parts of XSS. Next, requirements of the sanitization primitives from the perspective of real world applications in eight web applications are also analysed and found that there is a wide gap between the abstractions provided by the frameworks and application requirements.

Lomte V.M, et.al have presented different web attacks and provided some tricks used by hackers to hack websites and also mitigation techniques of these attacks in [14]. They have analysed two applications: with and without security, and also found the impact of Sql injection, XSS, DoS, and Request Encoding attacks in terms of request time, response time and throughput.

Chavan B, et al. have described in [15] the classification of the attacks and vulnerabilities that can affect website, its data or/and its users. These vulnerabilities are classified with respect to the phase of the development life cycle in which they arise. Vulnerabilities like Broken Access Control, authentication, Improper error handling, XSS, XSRF(Cross Site Request Forgery), Information Leakage, content spoofing, buffer overflow, injection related and many others have been presented. Also the countermeasures of the vulnerabilities and their weaknesses is discussed in tabular form.

Garg, A. et al. have presented in [16], five common and simple vulnerabilities in web applications. These vulnerabilities include Remote Code execution, SQL injection, Format String vulnerabilities, XSS, and username enumeration. Type of criticality and countermeasures of each vulnerability are also discussed.

Gadhiya, S.A, et. al. in [17], have presented research perspective of Cross site Scripting(XSS) Attack and its prevention in web applications. They have demonstrated various types of XSS attacks and also the mitigation techniques for it. It has been noted that preventing XSS attack is very difficult task and protection mechanisms also needed to be updated.

Singh, A.,et.al. , have surveyed the XSS attack in [18] and found that most recent attack on existing websites is DOM based. This attack can harm millions of people in few seconds as it exploits the vulnerabilities through submission method HTTP GET and HTTP POST. To prevent from this attack, a methodology of two way detector and filter is developed which identifies any suspicious URL submitted or stored in database of website and report to filter which is programmed to sanitize the data.

Duff, N. et. al., have presented in [19], an informational case study to the Cross Site Scripting attacks. Existing prevention approaches are discussed, which shows that a web developer should adopt secure programming practices in order to develop a secure website.

Kaur G. has shown XSS attack, its types and existing approaches to prevent these attacks in [20]. She has discussed the limitations of the existing techniques and proposed a new two – tier approach, one for detecting persistent and non-persistent XSS attacks and second for prevention of DOM based XSS attack.(Proposed technique is not implemented and tested.)

III. XSS VULNERABILITY AND ITS COUNTERMEASURES

Vulnerability is a weakness in the application which can be a design flaw or an implementation bug. An attacker can use such vulnerabilities, to harm the stakeholders of an application. SQL Injection Attack, Cross-Site Scripting (XSS), Cross- Site Request Forgery (CSRF), Broken Authentication and Session Management are some of the application layer vulnerabilities targeting most of the current web applications [21]. According to reports that are provided by OWASP [5-7] and WHID [22], among all these attacks SQLIA and XSS are very common. Also according to our previous research, XSS is one of the major attacks on web applications[23]. It is considered a severe of attack affecting confidentiality, integrity and availability of information. XSS attackers may use encoding, encryption, confusion and other technologies to evade the server-side filtering. The XSS Cheat sheet provides dozens of the encoding methods for evading the server-side filtering mechanism [27]. The current major browsers, such as the latest version of Internet Explorer, Google Chrome, Safari, Opera, etc. have provided the defense components of XSS attacks, but the realistic effects are not ideal. For instance, Internet Explorer 9 prohibit the cross-domain access as a default setting which prevents the JavaScript from one site sending POST requests to different sites. But this setting permit the JavaScript sending GET requests to any sites. As a consequence, an attacker can still steal the user's sensitive information as the GET request parameter.

As the main purpose of XSS attack is to execute malicious JavaScript in the victim's browser, and there are few fundamentally different ways of achieving that goal.

- **Reflected XSS:** In reflected XSS, the malicious string is part of the victim's request to the website. It might seem harmless as it requires the victim himself to actually send a request containing a malicious string. But attackers may trick the victims to send the malicious script without informing them. When the attacker targets a specific individual, group the attacker can send the malicious URL to the victim (using e-mail or instant messaging, or social networking link or link to his own site for example) and trick them into visiting it and stealing or changing their confidential information stored in cookies.
- **Persistent XSS:** In persistent/stored XSS, the malicious string originates from the website's database. It occurs when the data provided by the attacker is saved by the server, and then permanently displayed on "normal" pages returned to other users in the course of regular browsing, without proper HTML escaping. Here, malicious code is inputted by attackers into vulnerable web pages and is then stored on the web server for later use. The payload may be served back to other users browsing web pages and is executed in their context, at a later stage. Thus, the victims do not need to click on a malicious link in order to run the payload (as in the case of Non-Persistent XSS); they simply have to visit the vulnerable web page, serving back un-sanitized user input from other web sessions.

- **DOM-based XSS**: DOM XSS is a type of cross site scripting attack which relies on inappropriate handling, in the HTML page, of the data from its associated DOM. Among the objects in the DOM, there are several which the attacker can manipulate in order to generate the XSS condition, and the most popular, from this perspective, are the document.url, document.location and document.referrer objects.

XSS attacks have been around years now and lot of research in the field has been done by Industry and academic experts. In literature, there are many methodologies, algorithms and techniques proposed to provide a solution for prevention of XSS attacks. Analysis of XSS attacks reveal that they are caused due to improper coding of web applications and inability to filter or sanitize input and encode the output. So, here known XSS countermeasures and mitigation techniques from various researchers are classified in phases of SDLC.

Design Phase

- Use minimum text boxes and try radio buttons/drop down list/check boxes instead [25].

Coding Phase

- Santize/Validate Input by ensuring data is properly typed and does not contain escaped code.
- Validate inputs with Data Type, Data Length and Data Format at both client and server side.
- Encode string in such a way that all meta-characters are interpreted by the database as normal characters.
- For filenames, use stringent whitelists that limit the character set to be used. If feasible, only allow a single "." character in the filename to avoid weaknesses, and exclude directory separators such as "/". Use a whitelist of allowable file extensions.
- Use proper output encoding, escaping, and quoting. For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters.
- Do not trust client side input and enforce a tight check in the server side.

Testing Phase

- Conduct penetration tests against applications, servers and perimeter security[24].

IV. Various Attacks with XSS Vulnerability

All organizations who maintain a web presence are at risk of being attacked. However, the level of risk is different for each organization with respect to intellectual property or personally identifiable information stored by the organization [26]. The purpose of a web based attack is significantly different than other attacks. XSS is most commonly associated with execution of malicious javascript through web applications . This vulnerability is also utilized as a platform for launching other types of attacks. The other types of attacks include Session Hijacking, Denial of Service(DoS), Defacement, Account Hijacking and Cross Site Request Forgery(CSRF). For the case study of these attacks, a dummy web application is selected. Below described attacks are performed on this dummy web application manually. Test environment was created by installing XAMPP server on our local machine to make it a web server to host our dummy vulnerable application. Various attacks on the application with XSS vulnerability are performed and discussed here.

To verify the presence of XSS vulnerability a javascript code like

<script> alert("hi")</script>

is executed at the vulnerable parameter and it responds as shown in figure 1, which ensures the xss vulnerability existence in the page.

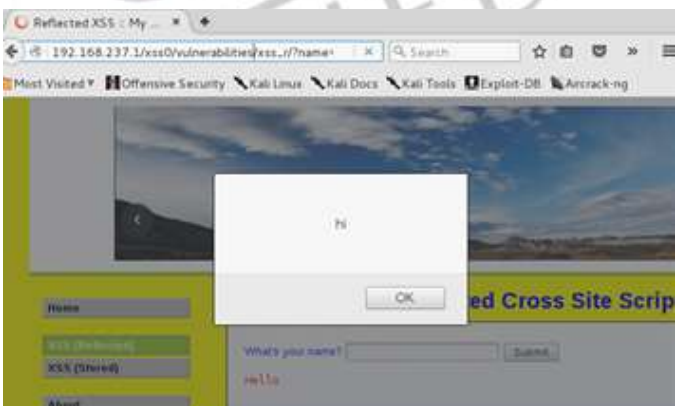

Figure 1: XSS vulnerability

When this vulnerability is given some other script to execute, it runs it with same efficiency and shows the confidential information which can be further used to steal the passwords and to perform other malicious activities. As Figure 2 shows the session ID and other information stored in the cookie.

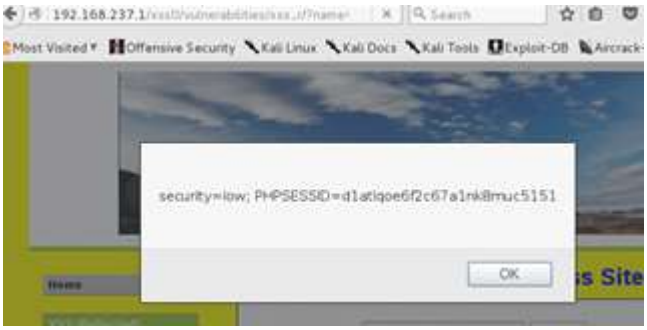

Figure2: Cookie values

1. Defacement: This is a very popular attack on the web and in this attack, an attacker alters the content of web site with offensive or erroneous graphics and/or text. An attacker can also change the appearance of the page or silently redirect a client to malware hosting server.

With Stored XSS vulnerability, the attacker is allowed to store the malicious script at the server which gets executed every time a user visits that page. But here, the script is such that which redirects the user to another page, and displays the page of another web site instead of the original page. Figure 3 depicts the javascript, which is to be stored at the server and figure 4 displays the result after the malicious script gets executed.

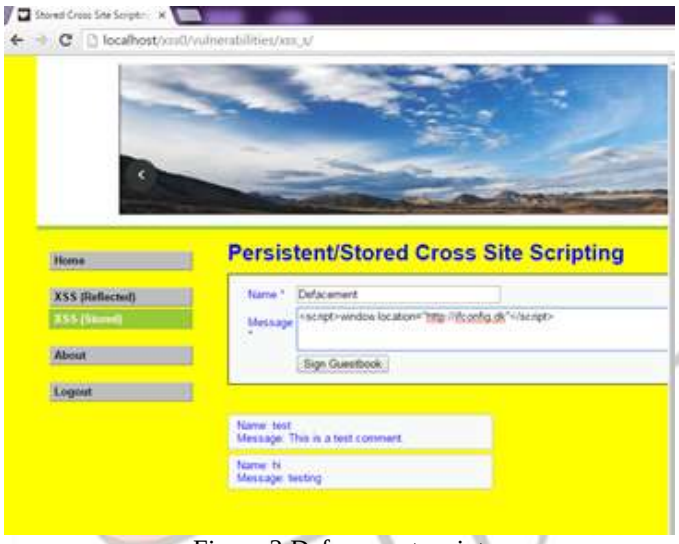

Figure 3:Defacement script

Figure 4:Defacement

2. Cookie Stealing: Cross Site Scripting vulnerability can be further exploited to steal the values stored in the cookies. It can be session ID of the user which in turn helps to hijack the user session.

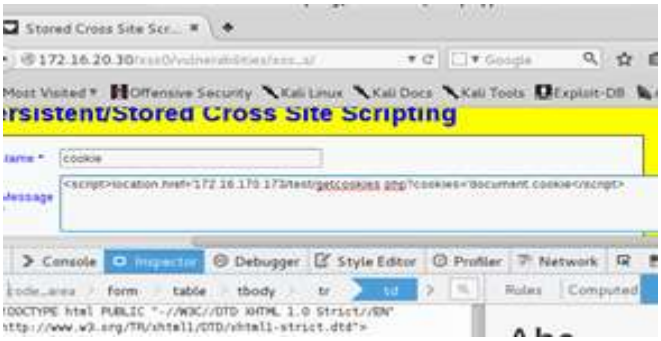

Figure 5:getcookies

Figure 5 shows the redirection of the victim's cookie to the malicious attacker's server or site without his knowledge. As the server has this stored XSS vulnerability and whenever any user visits its page, its cookies are transferred to the attacker's server. Attacker is continuously listening on the malicious server and gets the cookies value of each connected session as shown in figure 6.

Figure 6: Attacker Listening

By this way attacker can easily steal the cookies of the users and which in turn may be used by attackers to perform other harmful activities.

3. Session Hijacking: This is again a popular attack on the web and possible to perform with XSS vulnerability. In this attack, attacker uses the session ID received from the Cookies of the user to login without the requirement of the user account and password. Attacker changes the session ID using browser functionalities as shown in figure7, and replaces it with the one received as discussed before to login as that victim. By that way, attacker may easily make some changes to the data and even password of the user, which can further deny the victim to login.

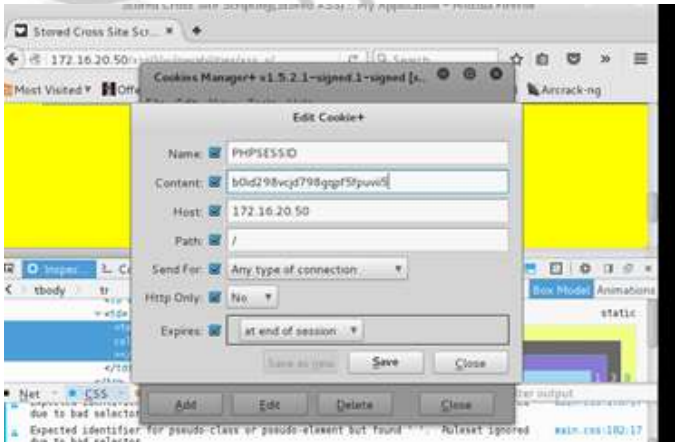

Figure 7: Replacing Session ID

This activity allows the attacker to login as the user and access his/her account information as figure 8 depicts the same.

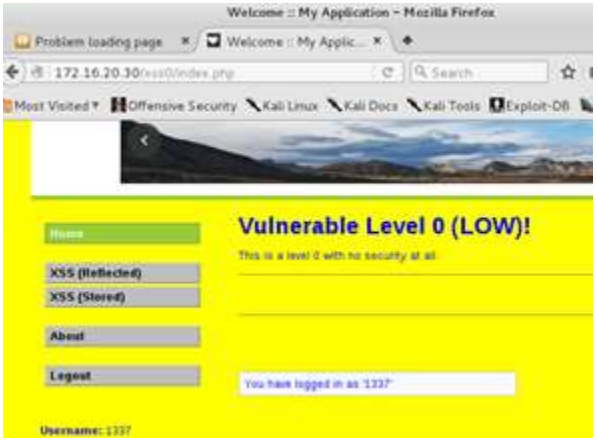

Figure 8: A/c login

4. Denial Of Service: This attack is also possible with XSS vulnerability as malicious javascript may deny the user to access the particular pages of the applications. The script visible in figure 9 reloads the page again and again, thus finally not let the users to access their pages.

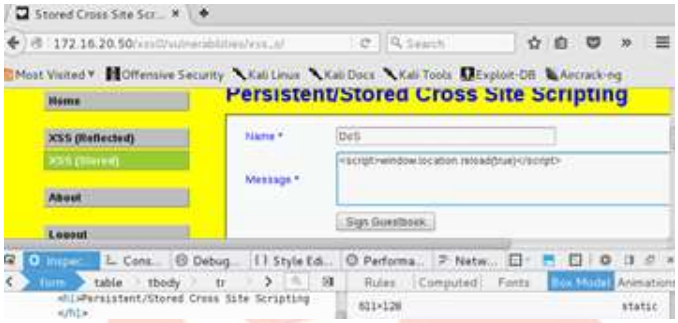

Figure 9:DoS with XSS

V. CONCLUSION

In this paper, we have studied and implemented the various attacks possible with XSS vulnerability in web applications. Thus the known countermeasures of this vulnerability are classified in the SDLC fashion and their effectiveness is checked with vulnerability scanner. Result of vulnerability scanner before and after the implementation of respective countermeasures reveal that if applications are developed with security in mind from the beginning of SDLC, then many attacks on web applications can be avoided almost without any extra effort and time. Thus only need of the time is to aware the developers with known countermeasures and their effectiveness. In future, other vulnerabilities and their corresponding attacks can be implemented to make web applications more safe and secure.

REFERENCES

[1]Abusaimeh, H. and Shkoukani, M. (2012). Survey of Web Application and Internet Security Threats. *International Journal of Computer Science and Network Security*. Vol 12, Issue 12, 67-76.

[2] Internet Security Threat Report, Symantec, vol.22, retrieved from: https://www.symantec.com/content/dam/symantec/docs/reports/istr-22-2017-en.pdf

[3] WhiteHat Website Security Statistics Report,2014. retrieved from https://www.whitehatsec.com/

[4] Web Application Attack Report,2015. Imperva. Retrieved from http://www.imperva.com/

[5] The Ten Most critical Web Application Security Risks, 2010. Open Web Application Security Project Top 10. Retrieved from http://www.owasp.org/

[6] The Ten Most critical Web Application Security Risks, 2013. Open Web Application Security Project Top 10. Retrieved from http://www.owasp.org/

[7] The Ten Most critical Web Application Security Risks,2017. Open Web Application Security Project Top 10. Retrieved from http://www.owasp.org/

[8] Joel Weinberger, Prateek Saxena, Devdatta Akhawe. A Systematic Analysis of XSS Sanitization in Web Application FrameWlrks. Springer-Verlag Berlin, Heidelberg. 2011. pp. 150-171,

[9] Lan,D., Ting,W.S.,Xing,Y. and Wei,Z. Analysis and prevention for cross-site scripting attack based on encoding,,IEEE Explore,2013.

[10]Kali Linux Documentation. Retrieved on 20 March,2016 from http://www.kali.org/

[11] Juillerat,N., Enforcing Code Security in Database Web Applications using Libraries and Object Models, LCSD 2007,Canada, ACM 1-58113-000-0/00/004.

[12]Shalini, S. and Usha S., Prevention of Cross-Site Scripting attacks (XSS) on Web Applications in the Client Side, International Journal of Computer Science Issues, Volume 8, Issue4, No.1, 2011.

[13] Weinberger, J. , Saxena, P. , Akhawe, D., Finifer,M., Shin,R. and Song,D., A Systemmatic Analysis of XSS Sanitization in Web Application Frameworks,ESORICS 2011
[14] Lomte, R.M and Bhura, S.A., Survey of different Web Application Attacks & Its Preventive Measures, IOSR Journal of Computer Engineering (IOSR-JCE), Volume 14, Issue 5. ISSN: 2278-8727
[15] Chavan, S.B and Meshram,B.B., Classification of Web Application Vulnerabilities, International Journal of Engineering Science and Innovative Technology (IJESIT),Volume 2, issue 2, 2013.
[16] Garg, A. and Singh, S., A Review on Web application Security Vulnerabilities, International Journal oF Advance Research in Computer Science and Software Engineering, Volume 3, Issue1, 2013.
[17] hiya,S.A and Wandra,K.H, The Research Perspective:XSS Attack and Prevention of XSS Vulnerability in Web Application, International Journal of Engineering Development and Research, volume2, Issue 4. ISSN:2321-9939.
[18]Singh, A. and Sthappan,S. , A survey on XSS web-attack and Defence Mechanisms, International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), Volume 4, Issue 3, 2014. ISSN:2277 128X
[19] Duff,N. and Yu, H. ,An Informational case study to Cross Site Scripting attacks, ??
[20] Kaur G., Study of Cross-Site Scripting Attacks and their Countermeasures, International Journal of Computer Applications Technology and research, Volume 3, Issue 10, 2014. ISSN:2319-8656
[21] M. Shema. “Seven Deadliest Web Application Attacks”. Elsevier Inc.,2010,47-69. ISBN-9781597495431
[22] Web Hacking Incident Database Project. Retrieved on 15 December,2015 from http://projects.webappsec.org/
[23]D. Kaur, P. Kaur. “Empirical Analysis of Web Attacks”. In Procedia of Computer Science. Elsevier Publications.DOI:10.1016/j.procs.2016.02.057
[24] W.K. Torgby, N.Y.Asabere.”Structured Query Language Injection (SQLI) Attacks: Detection and Prevention Techniques in Web Application Technologies”. International Journal of Computer applications Vol. 71-No.11 (May 2013). 29-40.ISSN: 0975-8887
[25] G. Parmar, K.Mathur. Proposed Preventive measures and strategies Against SQL injection Attacks”. Indian Journal of Applied Research, Vol.5, Issue 5(May 2015). 664-671. ISSN- 2249555X
[26] Web Based Attacks. Retrieved on 10 march from http://sans.org.reading-room/whitepapers/application/web-based-attacks-2053
[27] RSnake. XSS Cheat Sheet.Retrieved from http://hackers. org/xss.html