www.ijecs.in

*International Journal Of Engineering And Computer Science ISSN:2319-7242*

*Volume 3 Issue 9, September 2014 Page No. 8445-8450*

# A Study of Existing Cross Site Scripting Detection and Prevention Techniques in Web Applications

***Neha Gupta***[1]

[1] Kurukshetra University, Department of CSE,U.I.E.T

Kurukshetra, India

*ngupta514@gmail.com*

**Abstract:** ***Web Applications provide wide range of services to its users in an efficient manner. Web based attacks are increasing with the intent to harm the users or the reputation of particular organization. Most of these attacks occur through the exploitation of security vulnerabilities found in web applications. These vulnerabilities exists because developer focuses more on the development of the application rather than its security due to the time and budget constraints. Cross Site Scripting (XSS) is one of the major security vulnerability found in web applications. In 2013, XSS is ranked third among the top 10 list of attacks by OWASP (Open Web Application Security Project).XSS flaws occur whenever an application takes insecure data and sends it to the browser without proper validation or escaping. This can result in hijacking user session, defacing websites and redirecting the user to malicious sites. In this paper, we will study different existing techniques which can be used for detection and prevention of XSS attacks.***

**Keywords:** Cross Site Scripting, Web Application Security, Web Application Attacks, Security Vulnerabilities.

## 1. Introduction

Web Applications have become one of the most important ways to provide a broad range of services to users. In the recent years, web-based attacks have caused harm to the users of web applications. Most of these attacks occur through the exploitation of security vulnerabilities in the web-based programs. So, the mitigation of these attacks is very crucial to reduce its harmful consequences. Attackers can potentially use many different paths through your application to do harm to your business or organization or the users. A web page that is generated by the server and interpreted by the client browser has both text and HTML markup. Web sites with only static pages can have complete control over how their outputs are interpreted by the client. Web sites with dynamic pages cannot have full control over how the browser interprets these pages. The main issue is that if malicious content can be introduced into a dynamic web page, neither the web site nor the client is capable of recognizing that anything like this happened and prevent it. In 2013, XSS is ranked third among the top 10 list of risks by OWASP(Open Web Application Security Project)[16].

## 2. Cross Site Scripting

XSS (cross site scripting) flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. Cross Site Scripting allows an attacker to embed malicious scripts into a dynamic web page which is vulnerable, executing the script on user machine in order to gather data which can result in hijacking of user sessions, defacing web sites, or redirecting the user to malicious sites. A high level view of typical XSS attack is as shown in fig. 1[18]. Depending on the ways HTML pages reference user inputs, XSS attacks can be classified as reflected, stored, or DOM-based [17].

### 2.1 Reflected or Non Persistent XSS

These holes are present in a Web application server program where it references accessed user input in the outgoing web-page. This type of XSS exploit is common in error messages and search results. The malicious content does not get stored in server. Sever bounces the original input to the victim as shown in fig . 2[19].

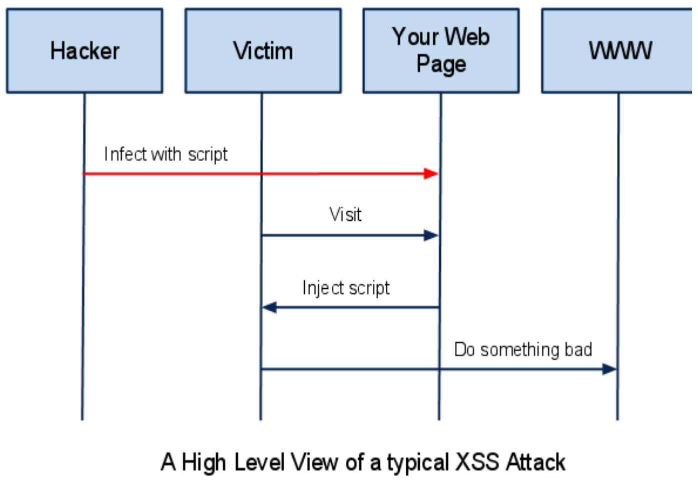


Figure 1: XSS Attack

### 2.2 Stored or Persistent XSS

These holes exist when a server program stores user input containing injected code in a persistent data store such as a database and then references it in a webpage. Attacks on social networking sites commonly exploit this type of XXS flaw. Server stores the malicious content and serves the content in original form as shown in fig. 3[19].

Both reflected and stored XSS holes result from improper handling of user inputs in server-side scripts.

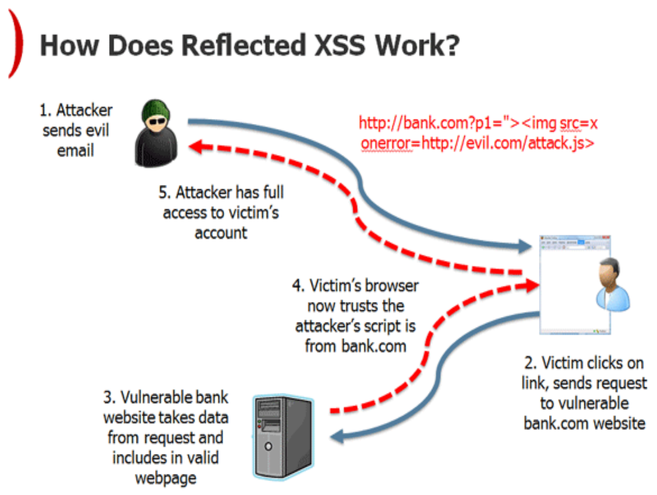

How Does Reflected XSS Work?
1. Attacker sends evil email
http://bank.com?p1="><img src=x onerror=http://evil.com/attack.js>
5. Attacker has full access to victim's account
4. Victim's browser now trusts the attacker's script is from bank.com
2. Victim clicks on link, sends request to vulnerable bank.com website
3. Vulnerable bank website takes data from request and includes in valid webpage

Figure 2: Reflected XSS

### 2.3 DOM Based XSS

In contrast, This is an XSS attack wherein the attack payload is executed as a result of modifying the DOM "environment" in the victim's browser used by the original client side script, so that the client side code runs in an "unexpected" manner. That is, the page itself does not change, but the client side code in the page executes differently due to the malicious modifications that have occurred in the DOM environment as shown in fig. 4.

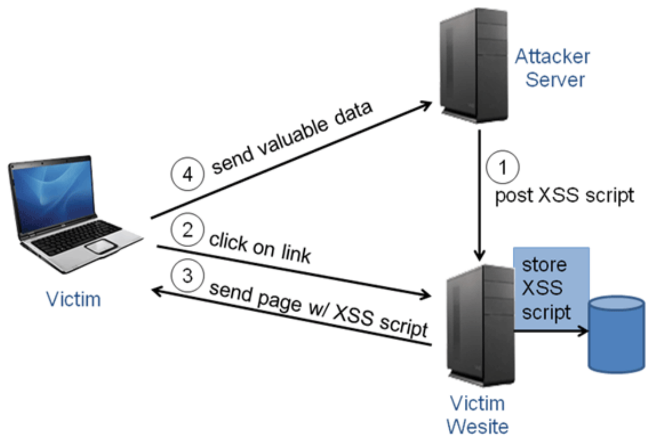
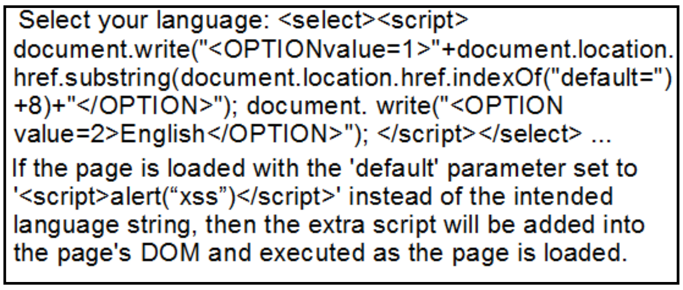


Figure 3: Stored XSS



```
Select your language: <select><script>
document.write("<OPTIONvalue=1>"+document.location.
href.substring(document.location.href.indexOf("default=")
+8)+"</OPTION>"); document. write("<OPTION
value=2>English</OPTION>"); </script></select> ...
```

If the page is loaded with the 'default' parameter set to '<script>alert("xss")</script>' instead of the intended language string, then the extra script will be added into the page's DOM and executed as the page is loaded.

Figure 4: DOM based XSS

## 3. Related Work

Over the past few years, there has been lot of research going on in both institutes as well as industries to prevent XSS attacks. Researchers have proposed some detection and prevention mechanisms discussed below:

[1]O.Ismail, M. Etoh, Y.Kadobayashi and S.Yamaguchi developed a proposal and implementation of automatic detection/collection system for cross site scripting vulnerability which is a client side system. It consists of detection/collection proxy server and a database server. For detection and collection on attack data two modes are used which are request change mode and response change mode. In request change mode if there are any HTML special tags in request or response, proxy will encode the tags and send the safe message. However, it does not work well if there are multiple parameters in request and response messages. We have response change mode for multiple parameters in which identifier will be attached with each special character. Thus system will send request with identity. After that all the collected information will be send to collection database server. It not only protects clients from XSS attacks but also inform the vulnerable web servers.

Some disadvantages with this approach are:

- How to use the collected information in database is not addressed.

- How to make system deployed universally has also not been addressed.

[2]T.Jim, N.Swamy and M.Hicks developed a mechanism that modifies the browser so that it can execute only legitimate scripts. It is based on two observations. First is that browser can perform script detection perfectly so the browser can be used to filter the scripts and second is that the developer of the web application knows scripts that should be executed for proper application functioning so the website can specify the legitimate scripts and filter the non legitimate scripts. In this the website embeds a security policy in its pages that specifies allowed scripts to run and browser enforce these policies i.e. security policies specifies what data the server sends to BEEP browsers. This mechanism requires minimal effort and low performance overhead. Also, it will prevent all the types of XSS attacks.

Some disadvantages with this approach are:

- It requires modifications in the frameworks or installation of additional frameworks.
- Approved scripts have to be identified by the website.

[3]Benjamin Liv-shits and Monica Lam developed a static analysis approach in which they used binary decision diagrams to apply points-to analysis to server-side script. This approach allows users to declare to specify information flow patterns succinctly and declaratively by describing a language called Program Query Language (PQL).It is static content sensitive and flow insensitive approach for information flow tracking analysis. It also has a model checking system which is used to generate the input vectors that expose the vulnerability. Thus with this approach we can find security vulnerabilities through the synergism of a new language which can be used for describing information flow, dynamic monitoring and model checking. It is easy to implement and adopt and can easily find the XSS vulnerabilities.

Some disadvantages with this approach are:

- It cannot check the correctness of input sanitization functions and, instead, generally assume that unhandled or unknown functions return unsafe data.
- It misses DOM-based XSS.
- It tends to generate many false positives.

[4]Motivated by static-analysis-based approaches Davide Balzarotti implemented a novel approach to analysis of sanitization process. More precisely a combination of static and dynamic analysis techniques can be used to identify faulty sanitization procedures that can otherwise be passed by an attacker. The static analysis component uses data flow techniques for identification of flow of input values from sources to sensitive sinks and the dynamic phase identify all the program paths from input sources to sensitive sinks that were identified during static analysis. If any malicious value reaches the sensitive sink during dynamic phase input is reported for violation of security. This approach is able to identify several vulnerabilities that form faulty sanitization procedures and it avoids false positives.

Some disadvantages with this approach are:

- The implementation only works on server-side scripts, so more research is needed on client-side script analysis.
- This approach can result in false negatives.
- It has incomplete attack string library because of the everyday introduction of new attacks.
- This technique suffers from state space explosion and thus might miss some vulnerability in deep state spaces.

[5]Siddharth Tiwari, Richa Bansal and Divya Bansal developed an optimized client side solution for cross site scripting which is a three step process i.e. script detector, analyzer, and data monitoring system. Every HTTP request will be passed to the script detector which reads the application level parameters and applies rules to the input i.e. it checks for the maximum number of characters. If there are more than the maximum number of characters in the input the input is rejected. Analyzer checks for the special characters in the request because for the execution of scripts input will be embedded with tags and special characters. If special characters exist it will be passed to the parser else request is processed. Analyzer uses whitelist and blacklist of sites maintained and synchronized with the server of security sites. Analyzer uses databases for detecting vulnerability. Data monitoring system monitors the flow of data. Operations that process sensitive data are marked along with results of those operations. If the marked data is to be transferred over the network user will be provided information about the consequences with the help of a dialogue box and then user well be asked to either allow or disallow the transfer. This approach is platform independent.

Some disadvantages with this approach are:

- It degrades the performance of client system.
- It requires client action.

[6]M.T. Louw and V.N. Venkatakrishnan developed a tool that works on existing browsers. The main objective of this approach is not to depend on browser's parser for building untrusted HTML parse trees. To accomplish this a parse tree is generated at server of the application with precautions that ensure that there is no dynamic content and the generated parse tree is then conveyed to document generator of the browser on the client browser without taking vulnerable paths. It is a tool that mitigates the XSS attacks with which first response pages are generated without any JavaScript node on server side, the removed script will be then executed on the browser side which is based on content generation by server side by code instrumentation. This ensures that all the unauthorized script execution will be prevented. It is effective

in most of the existing browsers, despite anomalous browser behavior and has acceptable performance overhead.

Some disadvantages with this approach are:

- It has to rely on external JavaScript library that is designed on client side.
- It also requires code instrumentation
- It requires installation of additional framework.

[7]E.Kirda et al developed Noxes which acts like a personal firewall that either allows or blocks connections to websites based on the filter rules, which are user-specified URL white lists and blacklists. Filter rules can be created manually in which user enters the set of rules in a database, or user can create a rule interactively whenever a request for connection is made which does not match an existing rule or user can use a snapshot mode in which Noxes tracks and collects domains that have been visited by the browser. It runs on the desktop of user as a background service and provides an additional layer of protection that was not supported by existing personal firewalls. When the browser sends a request to a website that is not known, Noxes alerts the client immediately, and asks the client to permit or deny the connection, and remembers the client's action for future reference. This approach covers all type of XSS attacks and clients don't have to rely on the web application for security.

Some disadvantages with this approach are:

- It requires client actions whenever a connection violates the filter rules.
- It only detects exploits that send user information to a third-party server, not other exploits such as those involving Web content manipulation.

[8]Hossain Shahriar and Mohammad Zulkernine developed MUTEC in which we apply the idea of mutation based testing technique to generate adequate data sets for testing the XSS vulnerabilities. In such technique an implementation is injected with faults to generate mutants. Rule for injecting fault is known as mutation operator. In this approach there are 11 mutation operators. A test case kills mutant if it causes different output between original program and the mutant. There are two mutant killing criteria. First is the number of HTML tags generated in DOM tree of implementation and mutant are not equal and second is HTML contents displayed in browser by implementation and mutant are not equal. The ratio of number of killed mutants to total number of non equivalent mutants is called mutation score and it can be used to measure the adequacy of test data set. This technique helps in discovering the vulnerabilities before the actual deployment.

Some disadvantages with this approach are:

- It requires intensive labor as the task of generating mutants is not automated.
- The effectiveness of testing based techniques depends entirely on the correctness of specification.

[9]P.wurzinger, C.Platzer, C.ludl, E.kirda and C.Kruegel developed a server side solution that detects and prevents cross site scripting attacks. SWAP includes a reverse proxy that intercepts all HTML responses and a modified browser which detects the script content. Proxy forwards the web response to a JavaScript detection component before it sends it to the client browser. All legitimate script calls in the web application are encoded in scriptIDs which are unparsable identifiers and so they are hidden from JavaScript detection component. If there are no scripts proxy will decode all scriptIDs and will deliver response to clients and if scripts are detected it notifies the client of XSS attack. This approach requires only simple automated changes of original web application and is able to distinguish between legitimate and malicious scripts.

Some disadvantages with this approach are:

- There is performance overhead.
- It is capable of detecting only JavaScript based attacks. It cannot defend against other malicious content.
- Also different web browsers may have different notation on valid and invalid JavaScript.

[10]Matthew Van Gundy and Hao Chen developed a mechanism to prevent cross site scripting exploits by enabling web clients to separate trusted and untrusted content by randomizing XML namespace prefixes in every document before serving it to the client. These randomized XML namespaces identify untrusted content and prevent such content from distorting the document tree. The web application partitions the web page content into different trust classes and a policy specifies the browser capability each trust class is allowed to exercise. It involves both server side and client side components. Server annotates every element and attribute of delivered document with trust classification and delivers the policy that specifies which elements, attributes and values are allowed for each trust class and the browser verifies whether parsed tree conforms to the policy. Noncepaces is simple and it does not need to sanitize untrusted content.

Some disadvantages with this approach are:

- It has moderate overhead.
- It does not protect against stored XSS attacks.
- For documents that are not based on XML, noncespaces does not work.

[11]Sid Stamm, Brandon Sterne and Gervase Markhan developed an approach that has content restrictions and content security policy. Content restrictions allow designers to specify content interaction on their websites which is basically a security mechanism needed by web and can be activated and enforced by web browsers when a policy is provided by HTTP. Content security policy specifies from where resources may be requested and the type of resources that may b loaded. It is effective to lock down sites and provide alert for vulnerabilities. Content security policy is activated by client browser when X-Content-Security-Policy HTTP header in

provided in HTTP response. The content of the header will either point to the file that contains policy or will directly state the policy. Content security policy can be used as an early warning mechanism for the attacks However, for each document, policies have to be specified in HTTP header.

Some disadvantages with this approach are:

- There is no single policy for all the documents.
- Creating policies manually is a very tough task.

[12]Sharth Chandra V. and S. Selvakumar developed BIXTAN which is a XSS sanitizer that is composed of HTML parser, a modified browser that acts as JavaScript tester and a mechanism for identifying static tags. When user enters code in a field of the web application, XSS sanitizer gets invoked. The user created HTML content is passed to the XSS sanitizer. XSS sanitizer will parse the content given by the user and during this HTML content is checked for static tags. Then static tags are retained and all other tags will be filtered. Though static tags do not invoke any dynamic content but some parsing quirks lead to invocation of dynamic content with use of static tags and to filter these quirks JavaScript tester is used. Static tags are then sent to JavaScript tester to check if there is JavaScript content. After the code is sanitized it is converted to DOM. Finally, a safe parse tree is generated when this code is returned to client. It is compatible with all the browsers and provides high fidelity and robustness.

Main disadvantage with this approach is:

- Browser source has to be modified for obtaining results.

[13]Rattipong Putthacharoen and Pratheep Bunyatnoparat developed a technique that is implemented in web proxy where cookies that are passed between user and web application are rewritten automatically. Basically, the name attribute in cookie will be rewritten automatically by a randomized value before it is sent to the browser database. Hence browser will have randomized value in its database rather than the value sent by the server. Cookie that will be returned by the browser will also be rewritten back to original value at web proxy before being forwarded to web server thus preventing cookie stealing. There is no change required on both browser and server.

Some disadvantages with this approach are:

- It has compatibility issues.
- It has performance overhead.
- There is a single point failure issue.
- This approach work only for HTTP's and does not work for SSL connections.

[14]Hossain Shahriar and Mohammad Zulkernine developed a server side approach which is based on boundary injection and policy generation notation. In this approach we pre and posted each dynamic content generation with a boundary which is a HTML or JavaScript content. Token is also inserted in each pair of boundary which is used to uniquely identify content generation or legitimate script location. Pair of boundary contains information on expected content features. This approach does not require understanding of whether suspected contents are derived from untrusted or trusted resources. Also, it does not need to transfer any sensitive information to the browser and it does not require code instrumentation. It also protects programs that suffer incorrect input filtering.

Some disadvantages with this approach are:

- This approach incurs runtime overhead due to interception of HTTP traffic.
- It requires user-defined security policies which can be labor-intensive.

[15]Takeshi Matsuda, Daiki Koizumi and Michio Sonoda developed a detection algorithm against cross site scripting attacks by extracting an attack feature of cross site scripting attacks and then considering the appearance position and frequency of symbols. It focuses attention on characters which are included in XSS attacks. It uses the idea of word extraction algorithm as a reference to construct the detection algorithm. It focuses attention on characters which are included in XSS attacks. It picked up 32 characters that are found in cross site scripting attacks. It basically finds position and frequency of those characters in input string to find the detection threshold. It detects cross site scripting attacks with help of attack feature value and threshold. In process of learning the threshold symbols found in cross site scripting attacks are ranked with numbers and other symbols are ranked as 0. From rank attack feature, value is computed against the input and based on this and input it is judged whether there is an attack or not.

Main disadvantage with this approach is:

- It requires the learning of detection threshold.

## 4. Conclusion

Cross site scripting has been a major threat for web applications and its users from past few years. Lot of work has been done to handle XSS attacks which include:

- Client side approaches
- Server side approaches
- Testing based approaches
- Static and dynamic analysis based approaches

Each kind of solution has been discussed in this paper. Different approaches have their own advantages and disadvantages. Major problems faced are:

- requirement of complex frameworks
- additional runtime overhead
- intensive labor requirements
- not being able to cover all types of XSS attacks

- prone to human error
- requires client action
- not able to detect web content manipulation
- false positives and false negatives
- effectives depend on completeness of specification

Based on our requirements we can choose among the possible solutions. However, there is no ideal solution for the detection and prevention of XSS attacks.

## References

[1] O.Ismail, M. Etoh, Y.Kadobayashi and S.Yamaguchi," A Proposal and Implementation of Automatic Detection/Collection System for Cross-Site Scripting Vulnerability," Proc. 18th IEEE International Conference on Advanced Information Networking and Application, 2004,pp.-145-151.

[2] T.Jim , N.Swamy and M.Hicks, " Defending against Cross-Site Scripting Attacks with Browser-Enforced Embedded Policies,"Proc of the WWW,Banff,Alberta,May 2007,pp. 601-610.

[3] M.S. Lam et al., "Securing Web Applications with Static and Dynamic Information Flow Tracking," Proc. 2008 ACM SIGPLAN Symp. Partial Evaluation and Semantics-Based Program Manipulation (PEPM 08), ACM, 2008, pp. 3-12

[4] D. Balzarotti et al., "Saner: Composing Static and Dynamic Analysis to Validate Sanitization in Web Applications," Proc. 29th IEEE Symp. Security and Privacy (SP 08), IEEE CS, 2008, pp. 387-401.

[5] Siddharth Tiwari, Richa Bansal, Divya Bansal, "Optimized Client Side Solution for Cross Site Scripting," IEEE 16th International Conference on Networks, December 2008, pp.1-4.

[6] M.T. Louw and V.N. Venkatakrishnan, "Blueprint: Robust Prevention of Cross-Site Scripting Attacks for Existing Browsers," Proc. 30th IEEE Symp. Security and Privacy (SP 09), IEEE CS, 2009, pp. 331-346.

[7] E. Kirda et al., "Client-Side Cross-Site Scripting Protection," Computers & Security,"Proc of 21st ACM Symposium on Applied Computing,Oct. 2009, pp. 592-604.

[8] H. Shahriar and M. Zulkernine, "MUTEC: Mutation-Based Testing of Cross Site Scripting," Proc. 5th Int'l Workshop Software Eng. for Secure Systems (SESS 09), IEEE, 2009, pp. 47-53.

[9] P.wurzinger,C.Platzer,C.ludl,E.kirda and C.Kruegel, "SWAP:Mitigating XSS Attacks using Reverse Proxy, "Proc. Of the SESS,Vancouver,Msy 2009,pp. 33-39.

[10] M.Gundy and H.Chen, "Noncespaces: Using Randomization to Enforce Information Flow Tracking and Thwart Cross-site Scripting Attacks," Proc. Of NDSS,San Diego,Feb. 2009.

[11] S.Stamm, B.Sterne and G.Markham, "Reining in the Web with Content Security Policy," Proc. of WWW, Releigh, North Carolina, April 2010, pp. 921-930.

[12] Sharath Chandra V. and S.Selvakumar, "BIXTAN:Browser Independent XSS Sanitizer for prevention of XSS attacks ,"ACM SIGSOFT ,September 2011, pp.1-7.

[13] R.Putthacharoen and P.Bunyatnoparat," Protecting Cookies from Cross Site Script Attacks Using Dynamic Cookies Rewritng Technique,"Proc. of IEEE 13th International Conference on Advanced Communication Technology, Feb 2011,pp. 1090-1094.

[14] Hossain Shahriar and Mohammad Zulkernine, "S2XS2: A Server Side Approach to Automatically Detect XSS Attacks ,"IEEE Ninth International Conference on Dependable,Automatic and secure computing,2011.

[15] Takeshi Matsuda , Daiki Koizumi and Michio Sonoda, "Cross Site Scripting Attacks Detection Algorithm Based on the Appearance Position of Characters"The 5th International Conference on Communications,Computers and Applications.Istanbul,Turkey,October 2012,pp.-65-70.

[16] Open Web Application Security Project,
Top 10 ,https://www.owasp.org/index.php/Top_10_2013-Top_10

[17] Cross Site Scripting Wikipedia,
http://en.wikipedia.org/wiki/Cross-site_scripting

[18] Cross site scripting
,accunetix,http://www.acunetix.com/websitesecurity/cross-site-scripting/

[19] Cross site scripting,Secure web development, ,http://hwang.cisdept.csupomona.edu/swanew/Code.aspx?m=XSS

## Author Profile

**Neha Gupta** pursuing the M.tech Degree in Software Engineering from University Institute of Engineering and Technology,Kurukshetra University in 2014.Doing Research on Detection and Prevention on Cross Site Scripting Attacks.