Cross Site Request Forgery: Preventive Measures

Sentamilselvan. K
Assistant Professor
Kongu Engineering College
Perundurai, Tamilnadu

Lakshmana Pandian. S
Associate Professor
Pondicherry Engineering College
Puducherry

Ramkumar. N
Assistant System Engineer
Tata Consultancy Services
Chennai

ABSTRACT
Cross Site Request Forgery is considered as one of top vulnerability in today's web, where an untrusted website can force the user browser to send the unauthorized valid request to the trusted site. Cross Site Request Forgery will let the integrity of the legitimate user. So far many solutions have been proposed for the CSRF attacks such as the referrer HTTP Header, Custom HTTP header, Origin Header, client site proxy, Browser plug-in and Random Token Validation. But existing solutions is not so immune as to avoid this attack. All the solutions are partially protected only. This paper focuses on describing the implementation of various possible cross site request forgery methods and describing the pitfalls in the various preventive techniques of cross site request forgery and so we suggested some defense mechanism to prevent this vulnerability.

Keywords
Security threats, Security breaches, Browser security, Forgery prevention, Defense mechanisms, Open web application security

1. INTRODUCTION
Now-a-days, Internet plays an important role for the business people and for the commercial use. Everyday life becomes easier for the internet users because of the progression in the technologies, but some vulnerability moves the web application to a risky environment. Even though many internet users get increased, the attackers too get increased in balance. So the security providence becomes must in the case of secure organization, defense personals and financial bank those interact with public. Aim of any companies is to provide a secure web service to their customers in the case of web environment and to safe guard the web from the threats. A report was submitted by open web application security project (OWASP) in the year 2013, on vulnerabilities based on critical web applications which can be demoralized [7]. From the survey among that Cross Site Request Forgery (CSRF) attack ranks 7th position. This attack is harsh in the case of web applications. Peter W introduced the general class of Cross Site Request Forgery attacks in a posting to the BugTraq mailing list most of the web application developer chose this attack [8].Most of the web developers do not have knowledge on Cross Site Request Forgery (CSRF) attack which is the common vulnerabilities among various attacks. In this attack victims are forced to perform an unwanted action on a trusted website, without any user's interaction [16]. Cross Site Request Forgery is an attack which forces an end user to execute unwanted actions on a web application in which he/she is currently authenticated [2]. With a little help of social engineering (like sending a link via email/chat), an attacker may force the users of a web application to execute actions of the attacker's choosing. A successful CSRF exploit can compromise end user data and operation in case of normal user. If the targeted end user is the administrator account, this can compromise the entire web application. Entirely single CSRF hole on the domain compromises the security. Reflected and stored are two types of attacks in which a malicious request, that is injected payload is hosted in a web page by a reflected CSRF other than a trusted website page.

Reflected XSS Attacks [9]: where the injected code is reflected off the web server, such as in an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request. Reflected attacks are delivered to victims via an e-mail message, or on some other web server. When a user is tricked into clicking on a malicious link the injected code travels to the vulnerable web server, which reflects the attack back to the user's browser. The browser then executes the code because it came from a "trusted" server. Therefore a victim is bared to an attack when they logs on to a trusted website and switch over to a different website concurrently. On considering stored CSRF attack the payload is present as part of a webpage downloaded from a trusted website [9]. This attack may found in blogs, forums, and message boards that frequently need a user to login before posting or viewing messages, where the injected code is permanently stored on the target servers, such as in a database, in a message forum, visitor log, comment field, etc. The victim then retrieves the malicious script from the server when it requests the stored information.

Login CSRF is one of the variations of Cross-Site Request Forgery, in which the attacker forges a cross-site request to the login form, logging the victim into the honest web site as the attacker. The severity of login CSRF vulnerability varies by site, but it can be as severe as a cross-site scripting vulnerability. Because of this attack private information are exposed. A valid HTTP request was provided by a CSRF attack which forces to perform an unwanted action in trusted website from browser. Based on the sensitivity operation which can be performed via these request will gives the harshness of the damage [1]. User profile, executing unauthorized financial transactions and so on are included in this. Authenticated users are tricked to perform a malicious action by an attacker, here the aim of attacker may get succeed because of the weakness in the design in the targeted application, cached credentials was automatically supplied by the users web site.

Overall diagrams give an idea of how CSRF attack was performed by an attacker. A malicious web site can force the user browser to send the unauthorized valid request to the targeted site. A user send a request to the server to view the needed web page based on the users request the web application was provided to the user by a server. While the user views here requested page a malicious script posted by an attacker was invisible or visible in the same page in the case iframe tag or image tag, but the user does not have a

knowledge that the link will direct to an unwanted page. Since the user does not well known about the link he may force to access the link by performing click operation on the malicious link. An automatic execution of malicious script takes place. As a result the attack will control the user's web page and send the forged request to the server via user's web browser. And the attacker will monitor the actions of a user repeatedly. This monitoring action was not known to the user but he thinks that he was in secure environment.

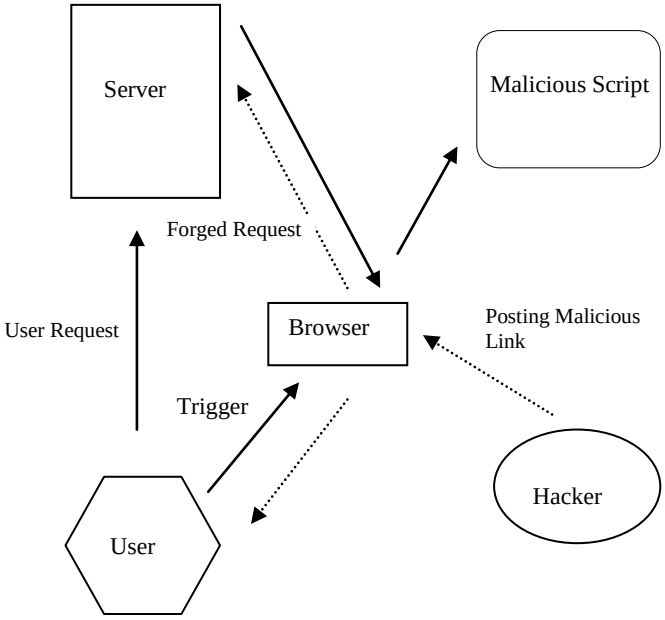

Fig1: Overview of CSRF Attack

2. LITERATURE SURVEY

Ramarao R, Radhesh M, Alwyn R Pais [9] was presented a client-side proxy solution that detects and prevents CSRF attacks using IMG element or other HTML elements which are used to access the graphic images for the webpage. This proxy is able to inspect and modify client requests as well as the application's replies (output) automatically and transparently extend applications with the secret token validation technique.

William Zeller and Edward W. Felten [11] implemented a client side browser plug-in that can protect users from certain types of CSRF attacks. They implemented their tool as an extension to the Firefox web browser. Users will need to download and install this extension for it to be effective against CSRF attacks. Their extension works by intercepting every HTTP request and deciding whether it should be allowed. This decision is made using the following rules. First, any request that is not a POST request is allowed. Second, if the requesting site and target site fall under the same-origin policy, the request is allowed. Third, if the requesting site is allowed to make a request to the target site using Adobe's cross-domain policy, the request is allowed. If their extension rejects a request, the extension alerts the user that the request has been blocked using a familiar interface (the same one used by Firefox's popup blocker) and gives the user the option of adding the site to a white list.

Sooel Son [10] was proposed PCRF is a dynamic token generating defense scheme against CSRF. PCRF's basic goal is to prevent CSRF attacks by adding a fresh token to every web request whose target page should be protected one way to efficiently prevent CSRF attacks toward PHP web applications. This defense system is called PCRF: Prevent Cross-site Request Forgery attack. PCRF provides an automatic robust solution again CSRF threats by using a CSRF token. Due to the property of cryptographically secure hash function, it used to verify whether the token has been previously issued from servers.

Nanad jovanovic et.al [6] proposed a mitigation mechanism for CSRF that provides only partial protection by replacing GET requests by POST requests or relying on the information in the Referer header of HTTP requests and also proposed a solution that provides a complete automatic protection from XSRF attacks. More precisely, his approach is based on a server-side proxy that detects and prevents CSRF attacks in a way that is transparent to users as well as to the web application itself (Orthogonal proxy).

Johns and Winter [4] introduced RequestRodeo, a client side solution to counter this threat. With the exception of client side SSL, RequestRodeo implements protection against the exploitation of implicit authentication mechanisms. This protection is achieved by removing authentication information from suspicious requests. They proposed a client side solution to enable security conscious users to protect themselves against CSRF attacks. Their solution works as a local proxy on the user's computer.

Tatiana Alexenko et.al [10] were developed a Mozilla extension that integrates with the Firefox web-browser to protect the user's browsing history. The extension generates HTTP requests to random URLs from the user's browsing history. The extension allows the user to specify how often the requests get sent as well as giving users the option of adding a random URL to the Referrer field of the extension-generated HTTP request. The latter option is bound to initiate discussion, because the pairing of the requested URL and Referrer is random which can lead to combinations that should not exist during normal browsing. This can affect online advertising and raise red flags for web administrators.

They implemented a client-side defense measure that previews the HTML code before each page load and detects potential CSRF attacks. The detector would first find all form tags and check the "action" attribute of the "form" tags for deep linking. If such forms are found, the CSRF detector will prompt the user if they want to add the pairing of the URL of the website the code is located on and the URL of the form action to a white list.

Burns and Schreiber [3] provide comprehensive introductions to CSRF attacks. To prevent the CSRF attack, they used following methods. Use cryptographic tokens to prove the Action Formulator knows a session specific secret, use secret tokens to prove the Action Formulator knew an Action and user specific secret, use the optional HTTP referrer [sic] header to verify Action Formulators, require changes to application state to be done only with HTTP POST operations and use a simplified CSRF Prevention Token.

Drawback of their proposed work is that the attackers can adjust their attacks to be form based like CSRF, Submit forms automatically or though tricking users by making huge, mislabeled submit buttons. The header is optional and may not be present, some browsers disable this header and it is not available when interactions occur between HTTPS and HTTP served pages. The risk of header spoofing exists, and tracking the valid sources of invocations may be difficult in some applications.

XSS vulnerabilities are being discovered and disclosed at an alarming rate [12]. XSS attacks are generally simple, but difficult to prevent because of the high flexibility that HTML encoding schemes provide to the attacker for circumventing server-side input filters. Paper describes an automated script-based XSS attack and predicts that semi automated techniques will eventually begin to emerge for targeting and hijacking web applications using XSS, thus eliminating the need for active human exploitation.

Any unauthorized user cannot find space in the communication. For client and server communication first client will be registered in the admin as an authorized node. Our algorithm can work on word, pdf and html types of mitigated files. If it is registered successfully then it can demand the file from server. Web-based attacks due to program security vulnerabilities are huge concerns for users. Efficient approach with DES encryption for better data receiving and sending mechanism was proposed [13].

A practical privacy-preserving approach was presented to defending against cross-site and same-site request forgery attacks. Fine-grained access control was used to allow a website owner to decide how requests should be sent and received within protection scopes, so as to prevent forged requests from being initiated outside the scopes. Two-phase checking as a building block that allows the browser and the website to exchange configuration information in a privacy-preserving manner was key process [14].

3. REAL TIME EXAMPLES OF CSRF ATTACK

3.1 ING Direct

It is one of the financial institutions where CSRF attack was first took place [11]. Attacks implemented by transferring the funds out of the user's bank account by the unauthorized people. This is due to the vulnerability on lNG's website. It makes to add the additional accounts on behalf of an arbitrary user.

3.2 Youtube

YouTube is one of the most viewable sites [11]. Vulnerabilities on the YouTube make the unauthorized people to add the account on behalf of the legitimate user. Using that account attacker added videos to a user's "Favorites," added himself to a user's "Friend" or "Family" list, sent arbitrary messages on the user's behalf, tagged videos as inappropriate, automatically shared a video with a user's contacts, subscribed a user to a "channel" (a set of videos published by one person or group) and added videos to a user's "Quick List" (a list of videos a user intends to watch at a later point)..

3.3 Metafilter

Metafilter is one of the site where vulnerability present here makes the user's account taken over by the attacker [11]. A forged request could be used to set a user's email address to the attacker's address. A second forged request could then be used to activate the "Forgot Password" action, which would send the user's password to the attacker's email address Equations..

3.4 The Newyork Times

Vulnerability present in the New York Time's website allows an attacker to find out the email address of an arbitrary user [11]. This takes advantage of the NYTimes's. Email this feature, which allows a user to send an email about a story to an arbitrary user. This email contains the logged-in user's email address. An attacker can forge a request to active the "Email this" feature while setting his email address as the recipient. When a user visit's the attacker's page, an email will be sent to the attacker's email address containing the user's email address. This attack can be used for identification (e.g., finding the email addresses of all users who visit an attacker's site) or for spam. This attack is particularly dangerous because of the large number of users who have NYTimes' accounts and because the NYTimes keeps users logged in for over a year. Also, Times People, a social networking site launched by the New York Times on September 23, 2008, is also vulnerable to CSRF attacks.

3.5 Gmail

In January 2007 this serious vulnerability was discovered in GMail which allowed an attacker to steal a GMail user's contact list [5].

3.6 Net Fix

It was discovered in Nettlix which allowed an attacker to change the name and address on the account, as well as add movies to the rental queue etc. [5].

3.7 G. EBay's Site

EBay is one of the auction sites, where more and more information get stored. CSRF attack was implemented which leads to loss of many personal information of about 18 million people. This issue was discovered in February 2008 [10].

4. PROPOSED WORK

4.1 Stored Attack

In the Stored attack, the attackers post the script in the server itself. This script was stored in the server. While user visits the page and doing any activities means in the script also gets execute and send information to the attacker. So the attackers know the information and user credentials. Here, we proposed pattern recognition method for preventing the stored attack.

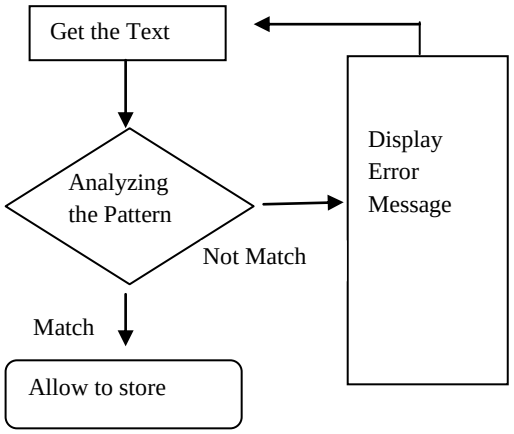

Fig 2: Implementation of Stored Attack

In this method, if the text contents matches the following regular expression for the "regex=/^[a-zA-Z0-9 .,:;{}!@#$%_?^*()<]+$/;" and "regex=/<imgsrc=|<form|<iframe=|<a ref=|<script=/;" pattern means it will allow to post the comment and it will be stored in server. Otherwise it will not allow to posting the comments. Using this regular expression pattern we can avoid the stored attack.

4.2 Login CSRF Attack

In the Login CSRF attack, Attacker and honest user are considered as an authenticated user for a provided web service. In general the username and password entity present in the URL remains same for all users with certain slight entity variation in it. Attacker will generate a malicious link using his username and password and include it in the user's web page which may be visible for them.

If the user clicks the malicious link, he will enter into the attacker account without user's knowledge. Then the attacker enters into his login and to view the users account history. Now attacker feels free to steal the user's identity, or to spy on the user. Normally the user authentication was performed based on the username and password. This user credential will be stored in the database. If the wrong authentication gives a negative result an error message will be displayed. This attack was not prevented by username and password verification process because the attack was carried out by an attacker using the username and password. To prevent the login CSRF attack a secure 2-step verification method was enhanced. In this method a random number generation was carried out by a server and sends it to the user's mail or mobile, using the provided code the user will prove him as an authenticated user. Using that code user can view his homepage

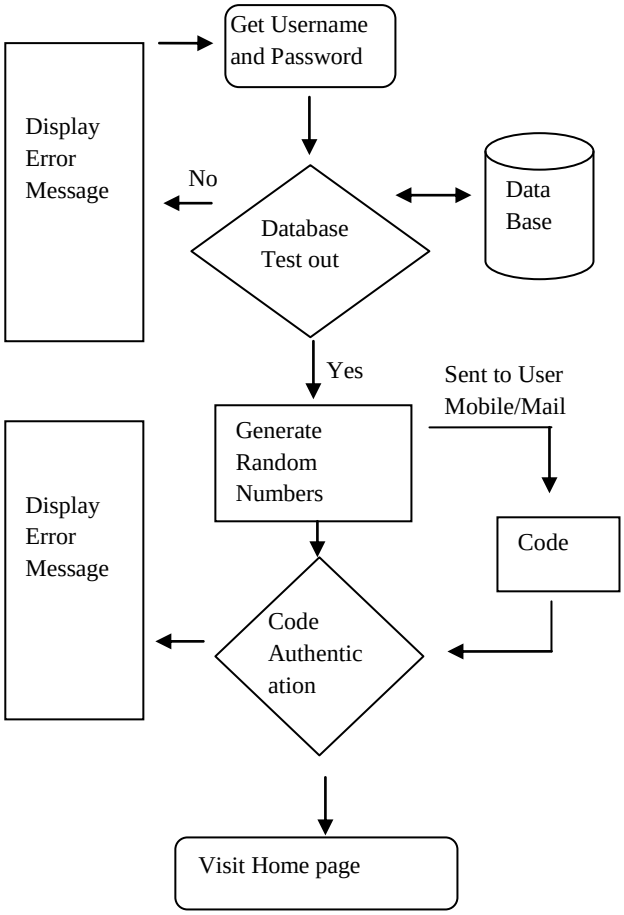

Fig 3: Login CSRF Attack

5. EXISTING DEFENCE MECHANISMS

Since CSRF attack is one the Vulnerability which presents in the web applications. It is need to be given mitigation, so that information present will be safeguarded. Below are the Defenses Mechanisms, which can able to protect the information to some extent. It is a policy that needs to be adopted by developers and users to avoid this attack to some extent.

5.1 Using Random Token

To defend against the CSRF attack it is one of the greatest mechanisms. Form proposal was done during the use of random tokens at all time. To fill in malicious URL, prediction of next random pattern was difficult for the aggressor [5].

5.2 Using of Form Post

Form submission involves two methods they get and post. Form submission is the secure one for post method. Variables and values in URL are understood by anyone in get method as a query strings [5].

5.3 Limitation the Lifetime of Authentication Cookies

Beside CSRF it is a durable deterrence. In a short period of time lifetime was limited. After a short period of time cookies will be terminated when the user moves to further web site. For any action user involves in re-login. Re-submission of password by the user was not done if the attacker tries to send any HTTP request [5].

5.4 Damage Limitation

To reduce the damage from CSRF those steps are followed by the Destruction restriction. To perform CSRF by an attacker on a website an authentication was required for every usage to limit the damage [5].

5.5 Forcing the User to use the Form

Force the user always to use the form of website. For this purpose a hidden fields are used which a helpful one. It is one of the protection and easy to bypass [5].

5.6 Auto Logoff

If a user moves to some other site (untrusted) means it will automatically log off. So again the user wants to login. Don't start new task while sensitive task running: If the user is using sensitive task means don't start new application or task (untrusted).

6. PERFORMANCE METRICS

Table 2 comprises of the performance metrics which is done by using some sample cases in real time environment.

6.1 Sample Test Case

Table 2. Performance Metrics done in Real time Environment

TRAIL CASES	ACCEPT/ REJECT
Hai	Accepted
GOOD MORNING	Accepted
Thank you….	Accepted
How are you?	Accepted
abc@gmail.com	Accepted
<imgsrc=a.jpeg>	Rejected
	Rejected
https://www.facebook.com/groups/38749/42826355/?ref=notif&notif_t=group_activity	Accepted
www.google.com	Accepted
<html>	Rejected
<a>	Rejected
	Rejected
<h1>	Rejected
my mobile no.+91-9894098940	Accepted
abc_xyz@yahoo.co.in	Accepted
Wow…….!!!!!!!!!!!!!	Accepted
$ 400 rs	Accepted
☺	Accepted
☹	Accepted
<3	Accepted
:'(	Accepted
100%	Accepted
{hbnbgkj}	Accepted
[67]	Accepted
(note: 12346)	Accepted
5/8	Accepted
7>8	Accepted

Table 3 : System Evaluation Result

True accept	False accept	True reject	False reject
207	0	90	0

Here, we have taken around 300 trails and we have analyzed from that result. From this analysis we got 208 true accept, 0 false accept, 90 true reject and 0 false reject.

TRUE ACCEPT-The texts do not have any malicious script and it is accepted

TRUE REJECT-The text contain with malicious script and it is rejected.

FALSE ACCEPT-The text contains with malicious but it is accepted.

FALSE REJECT-The text do not have any malicious script but it is rejected.

7. ADVANTAGES AND DISADVANTAGES

The following tables give knowledge about the advantages and disadvantages for the existing solution [17].

Table 3. Advantages and Disadvantages

Existing Solution	Advantages	Disadvantages
Browser Plug-in	It's Simple	It may sometime Crushed. May be Some user aware of this plug-in.
Clint Side Proxy	It is easy way to monitor and find attack.	If proxy Compromised means all sensitive information will lost. It won't detect the login CSRF
Secret Token Validation	Computational is low	Requires Dynamic generation
Random Validation Token	It is one of the best solutions. Simple to implement.	It needs SSL to be implemented in all applications. It won't Detect Login CSRF
Cryptographic token	Very Strong Protection and it requires no additional memory	Requires Dynamic Generation and requires a small amount of system resources to check tokens and big database tables to manage tokens and sessions.
Referrer Header	Simple to	Many browsers

	implement	disable this Header
Origin Header	It one of the way to find the same site request and cross site request.	Many browsers disable this Header
Origin Header	It one of the way to find the same site request and cross site request.	Many browsers disable this Header
Custom Header	It also one of the way to find the same site request and cross site request.	Many browsers disable this Header
Captcha[16]	Easy to Store in the memory. It is better Solution for auto submitting forms.	Requires more memory. It is expensive
Code Verifier	It will detect the login CSRF	It will take long time to verify the code

8. CONCLUSION

Cross Site Request Forgery is one of the top vulnerabilities in the internet. It remains challenging for the researchers to provide a better solution for mitigating this attack. There were many organizations which affected by this cross site request forgery attack. Defense mechanisms and existing solutions for cross site request forgery are working in some extend only. The above work can be extended to provide suitable solutions for the cross site request forgery attack by means of applying parsing techniques to identify the attacking spots before the attackers attack. Some pattern for img, script, form, iframe tags can be designed to identify the attack.

9. REFERENCES

[1] A.Barth, C.Jackson, and J.C.Mitchell. "Robust defenses For cross site request forgery". In Proc. ACM Conference on Computer and Communications Security (CCS), Oct, 2008.

[2] Cross-Site Request Forgery. www.owasp.org/index.php/CrossSite_Request_Forgery, May, 2009.

[3] J. Burns. Cross Site Reference Forgery: An introduction to A common web application weakness. http://www.isecpartners.com/documents/XSRF_Paper.pdf, 2005.

[4] M. Johns and J. Winter, "RequestRodeo: Client Side Protection against Session Riding," In Proc. of the OWASP Europe Conference, Leuven, Belgium, May 2006.

[5] Mohd. Shadab Siddiqui and Deepanker Verma,"Cross Site Request Forgery: A common web application weakness",

IEEE Conference and white paper, 2011.

[6] Nenad Jovanovic, Engin Kirda, and Christopher Kruegel.

"Preventing cross site request forgery attacks".In IEEE International Conference on Security and Privacy in Communication Networks (SecureComm), 2006.

[7] OWASP. Top ten most critical web application security vulnerabilities.https://www.owasp.org/index.php/Top_10_2013-Top_10.Forgeries.www.securityfocus.com/archive/1/19S90,2001.

[8] Ramarao R. Tool "preventing image based CSRF attacks". http://isea.nitk.ac.in/rod/csrf/PreventImageCSRF/. May, 2009.

[9] Sooel Son, "Prevent Cross site Request Forgery PCRF"userweb.cs.utexas.edu/~samuel/PCRF/Final_PCRF_paper.pdf.

[10] Tatiana Alexenko Mark Jenne suman Deb Roy and Wenjun Zeng," Cross-Site Request Forgery: Attack and Defense". In Proc. IEEE Communications Society (CCNC), 2010.

[11] W. Zeller and E. W. Felten, "Cross-Site Request Forgeries: Exploitation and Prevention," Technical Report, Princeton University, 2008.

[12] E. Kirda, C. Kruegel, G. Vigna, and N. Jovanovic. Noxes, "A Client-Side Solution for Mitigating Cross Site Scripting Attacks", Proceedings of the 21st ACM Symposium on Applied Computing, 2006.

[13] Sapna Choudhary, Bhupendra Singh Thakur, "DES Encryption and Attack detection in Client-Server Communication", International Journal of Advanced Research in Computer Science and Software Engineering. Volume 4, Issue 3, March 2014.

[14] B. S. Y. Fung, "A Fine-Grained Defense Mechanism against general Request Forgery Attacks", In Proc. of IEEE/IFIP DSN Student Forum, 2011.

[15] Luis von Ahn, Nick Hopper Manuel Blum, and John Langford, "CAPTCHA: Using hard AI problems for security", In Eurocrypt 2003.

[16] Sentamilselvan K, S Lakshmana Pandian, Dr.K.Sathiyamurthy. "Survey on Cross Site Request Forgery." IEEE International Conference on Research and Development Prospects on Engineering and Technology (IEEE ICRDPET-2013). Vol. 5. No. 5. IEEE, 2013.

[17] Sentamilselvan K, Prasath T. "A conceptual study of Cross Site Request Forgery with comprehensive scrutiny." International Research Journal Of Sustainable Science & Engineering 1.ISSN: 2347-6176 Issue:1 (2014): 1-6.