International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-3 Number-1 Issue-9 March-2013

An Efficient Partition Technique to reduce the Attack Detection Time with Web based Text and PDF files

Animesh Dubey[1], Ravindra Gupta[2], Gajendra Singh Chandel[3]

Abstract

In this paper we propose an efficient partition technique for web based files (jsp, html, php), text (word, text files) and PDF files. We are working in the direction of attack time detection. For this motivation we are considering mainly two factors first in the direction of minimizing the time, second in the direction of file support. For minimizing the time we use partitioning method. We also apply partitioning method on PDF files. In this paper we proposed an efficient hybrid approach which is the combination of Key generation, partitioning and encryption. In this approach admin register the client after the connection request from the client. Then client can approach for the data. Admin now call the data preparation strategy for data preparation for sending. In the preparation stage admin first generate the key which is common for server and client, then partition the data, so that the file overhead is reduced and encrypt the file by java default encryption method. After successfully preparation, the data log will be maintained and the data will be sending to the client by adding a hidden bit. When any attacker updates the data the hidden bit is automatically changes and an alert will be generated to the server. Server record the time of alert and inform to the client by changing the hidden bit, client understands the attack by checking their log file. If the hidden bit is changed in the client side then client understand the attack and send the request again for the same file. In the result section we provide the comparison with the traditional technique which shows the effectiveness of our approach.

Keywords

Partition, Hidden Bit, Content Sniffing, Encryption.

Animesh Dubey, M.Tech Scholar, CSE, SSSIST, Sehore, Bhopal, India.
Ravindra Gupta, Assistant Professor (CSE/IT), SSSIST, Sehore, Bhopal, India.
Gajendra Singh Chandel, HOD (CSE/IT), SSSIST, Sehore, Bhopal, India.

1. Introduction

If we analyze our daily routine, then we surprise to observe that we are relying on Internet. It is our need. For example email, e-shopping, trading, game etc. In the meantime we share our crucial and confidential data by HTML browser. Most of the data we share is text files, doc file, PDF file and Images. So we are very much concern on the security issue when we exchange the data from source to destination. We can understand the phenomena better in terms of client and server side data exchange. Sending data usually reside on a server-side and are accessed from its client-side [17]. There are some approaches which is either applied on client side as well as the server side but overall the approaches are not well enough to protect with the vulnerabilities. As a result users are fear and sometimes he/she may be suffering from those vulnerabilities [17].

If we understand the above scenario then we better understand and realize the unwanted security threats. personal information for example phishing websites [1] instead of providing legitimate functionalities. Thus, the mitigation of web-based security vulnerability exploitations is extremely important to reduce some of the consequences. Phishing is a rapidly growing problem, with 9,255 unique phishing sites reported in June of 2006 alone [2]. It is unknown precisely how much phishing costs each year since impacted industries are reluctant to release figures; estimates range from $1 billion [3] to 2.8 billion [4] per year.

For this reason we study a number of common program security problems and vulnerabilities [5][6]. Our study focuses that the number of web-based attacks has increased in recent years [7][8], existing research has addressed a subset of security vulnerabilities in web applications for example SQL Injection. Some security encryption technique like RSA is also suggested in [9]. So data security is the important concern [10][11][12]. After observation from several research by different authors, we analyze there are several numbers of vulnerabilities are still in the communication process when we want to access data from the web. So in this paper we

want to survey the aspects of content sniffing attacks. What are the major precautions considered by different authors with their pros and cons are discussed. The remaining of this paper is organized as follows. In Section 2 we discuss about content sniffing attack. The related work in section 3.In section 4 we discuss about problem domain. In section 5 we discuss the analysis. The conclusions and future directions are given in Section 6. Finally references are given.

2. Literature Review

In 2010, Hossain Shahriar et al. [13] discuss about Cross Site Request Forgery (CSRF) which allows an attacker to perform unauthorized activities without the knowledge of a user. An attack request takes advantage of the fact that a browser appends valid session information for each request. As a result, a browser is the first place to look for attack symptoms and take appropriate actions. According to the author Current browser-based detection methods are based on cross-origin policies that allow white listed third party websites to perform requests to a trusted website. To alleviate these limitations, they present a CSRF attack detection mechanism for the client side. Their approach relies on the matching of parameters and values present in a suspected request with a form's input fields and values that are being displayed on a webpage (visibility). To overcome an attacker's attempt to circumvent form visibility checking, they also compare the response content type of a suspected request with the expected content type.

In 2011, Misganaw Tadesse Gebre et al. [14] proposed a server-side ingress filter that aims to protect vulnerable browsers which may treat non-HTML files as HTML files. Their filter examines user uploaded files against a set of potentially dangerous HTML elements (a set of regular expressions). The results of their experiment show that the proposed automata-based scheme is highly efficient and more accurate than existing signature-based approach.

In 2011, Anton Barua et al. [15] developing a server side content sniffing attack detection mechanism based on content analysis using HTML and JavaScript parsers and simulation of browser behavior via mock download tests. They have implemented our approach in a tool that can be integrated in web applications written in various languages. In addition, they have developed a benchmark suite for the evaluation purpose that contains both benign and malicious files. They have evaluated our approach on three real world PHP programs suffering from content sniffing vulnerabilities. The evaluation results indicate that their approach can secure programs against content sniffing attacks by successfully preventing the uploading of malicious files.

In 2012, Usman Shaukat Qurashi et al. [16] discusses about AJAX (asynchronous JavaScript and XML) based attack. According to the authors an AJAX enabled web application is composed of multiple interconnected components for handling HTTP requests, HTML code, server side script and client's side script. These components work on different layers. Each component adds new vulnerabilities in the web application. The proliferation AJAX based web applications increases the number of attacks on the Internet. These attacks include but not limited to CSR forgery attacks, Content-sniffing attacks, XSS attacks, Click jacking attacks, Mal-advertising attacks and Man-in-the-middle attacks against SSL etc. Current security practices and models are focus on securing the HTM. They focus on addressing security issues observed in AJAX and Rich Internet Applications (RIA) and compiling best practices and methods to improve the security of AJAX based web applications.

In 2012, Syed Imran Ahmed Qadri et al. [17] provide a security framework for server and client side. In this they provide some prevention methods which will apply for the server side and alert replication is also on client side. Content sniffing attacks occur if browsers render non-HTML files embedded with malicious HTML contents or JavaScript code as HTML files. This mitigation effects such as the stealing of sensitive information through the execution of malicious JavaScript code. In this framework client access the data which is encrypted from the server side. From the server data is encrypted using private key cryptography and file is send after splitting so that we reduce the execution time. They also add a tag bit concept which is included for the means of checking the alteration; if alteration performed tag bit is changed. Tag bit is generated by a message digest algorithm. We have implemented our approach in a java based environment that can be integrated in web applications written in various languages.

In 2012, Sudhakar Parate et al. [18] provides a comprehensive review on various techniques that helps to improve the system, analysis of different attack, detection of attack. It focuses on the critical stages of preservation and acquisition of digital evidence from the different source to be used as evidence for aiding investigation.

3. Proposed Work

In this paper we have proposed a secure server client environment for detecting content sniffing attack. Our attack detection Framework works on below file formats:
1) Text
2) HTML
3) PHP
4) PDF
5) Java Script

We explain our efficient methodology in the subsequent sections. In this framework first admin register the client so that client is authorized to access the data. Then client request with the appropriate file which he/she needed. Admin receives and acknowledge the request. Then admin prepare the data for send. Admin data preparation takes four steps:

Key generation: In this phase a random key generation technique is applied from the server side. It will be generated every time when a fresh request is arrived by the user.

Partition data: For reducing the overhead we partition the data, so that the overhead of the files will be maintained.PDF Splitting is not achieved in [17]. We achieve page wise splitting of PDF files, so that the time will be reduced.

Encryption data: Admin first encrypt the data by Private Key cryptography, which uses the same key to encrypt and decrypt the message. This type is also known as symmetric key cryptography. In java we can use Base64 encryption and decryption as defined by RFC 2045 which provide a symmetric key encryption. With symmetric encryption, both parties use the same key for encryption and decryption purposes. Creates a Base64 codec used for decoding (all modes) and encoding in the given URL-safe mode.

Hidden Bit: Then the admin add a hidden bit with the encrypted file for the security of the file. If the file is attacked or alters by any malicious attacker then it is automatically changed and an alert will be replicated to the server. Then the data send to the client. The whole process is shown in figure 5. We have implemented our approach in a java based environment that can be integrated in web applications written in various languages. We develop server-side attack detection frameworks to detect attack symptoms within response pages before sending them to the client. The approaches are designed based on the assumption that the server-side program source is available for analysis, but we are not allowed to alter the program code after sending in the client environment. Admin of the server first authorize a client to connect to the server. So that proper log file of authorize client should be maintained. Then client can request only supported files from the server, in our case supported files are web pages, text data and PDF. Otherwise request is not granted as shown in Figure4. Then admin generates key for decryption, by using key generation algorithm as shown in figure 1. Then we apply partition and default encryption scheme of java as shown in figure 2 and figure 3. As per the above algorithm if the size is less than or equal to100 KB it is partition into 2 parts, if it is less than or equal to 250 KB then it is partition into 3 parts, if it is less than or equal to 500 Kb then it is partition into 4 parts, otherwise it is partition into 6 parts. PDF split page wise. Then From the server data is encrypted using private key cryptography and file is send after splitting so that we reduce the execution time. We also add a hidden bit concept which is included for the means of checking the alteration; if alteration performed the hidden bit is changed. This approach provides the security in the server side and alert the client which reduces the non-secure violation with data use. In this approach client want to establish a secure connection from the server for gathering data from the server. Client simply requests the data and the admin provides the available resources from the server database. Each user must possess the same key to send encrypted messages to each other. The sender uses the key to encrypt their message, and then transmits it to the receiver. The receiver, who is in procession of the same key, uses it to decrypt the message. The security of this encryption model relies on the end users to protect the secret key properly. If an unauthorized user were able to intercept the key, they would be able to read any encrypted messages sent by other users. It's extremely important that the

users protect the key themselves, as well as any communications in which they transmit the key to another person. For time detection we also open the link for the attacker so that we check the time of attack. If the attacker attacks and successfully updates the content then the hidden bit will be changed automatically and an alert response will generated at the server side. Servers maintain the attack time and replicate the time to the client and notify the client for the attack by the hidden bit. In this manner we can detect the attack. The analysis of detection is explained in the result analysis section to show the effectiveness of the algorithm.

Key Generation Algorithm

Step 1: Random random = new Random();
Step2:Strings1=new String("abcdefghijklmnopqrstvuwxyz");
Step3:Strings2=new String("ABCDEFGHIJKLMNOPQRSTVUWXYZ");
Step 4:String s3=new String("0123456789");
Step 5: int r1 = random.nextInt(26);
Step 6:String key=new String();
Step 7: key=String.valueOf(s1.charAt(r1));
Step 8:r1 = random.nextInt(26);
Step 9:key=key+String.valueOf(s2.charAt(r1));
Step 10:r1 = random.nextInt(10);
Step 11:key=key+String.valueOf(s3.charAt(r1));
Step 12:r1 = random.nextInt(26);
Step 13: key=key+String.valueOf(s2.charAt(r1));
Step 14: r1 = random.nextInt(26);
Step 15: key=key+String.valueOf(s1.charAt(r1));
Step 16: r1 = random.nextInt(10);
Step 17: key=key+String.valueOf(s3.charAt(r1));
Step 18: return(key);

Figure 1: Key Generation Algorithm

Partition Algorithm

Step 1: Initialization
int c=0; // counter is initializes to 0
int len=0; // Length of file is initializes to 0
Step 2: File f=new File(f1);
Step 3: long size=f.length()/1024;
Step 4: if(size<=100)
len=(int)f.length()/2;
Step 5: else if(size<=250)
len=(int)f.length()/3;
Step 6: else if(size<=500)
len=(int)f.length()/4;
Step 7: else
len=(int)f.length()/6;

Figure 2: Partition Algorithm

By the below combination we ready the data for preprocessing, so that it is secure. If any way attacker attacks then we add the hidden bit to detect, so that by detection the client can re-request the file for again preprocessing the data. So by approaching the flowchart(figure 4) we can secure the data or alert the server for the attack.

Encryption Algorithm

Step 1: Cipher cipher = Cipher.getInstance(xform);
Step2: IvParameterSpec ips = new IvParameterSpec(iv);
Step3:cipher.init (Cipher.ENCRYPT_MODE, key, ips);
Step 4: return cipher.doFinal(inpBytes);
Step 5: private static byte[] decrypt(byte[] inpBytes, SecretKey key, String xform)
Step 6: Repeat step 1 and step 2.
Step7:cipher.init(Cipher.DECRYPT_MODE, key, ips);
Step 8: return cipher.doFinal(inpBytes);
For the above algorithm we use java packages like crypto.KeyGenerator,crypto.SecretKey,crypto.spec. IvParameterSpec and crypto.Cipher.

Figure 3: Encryption Algorithm

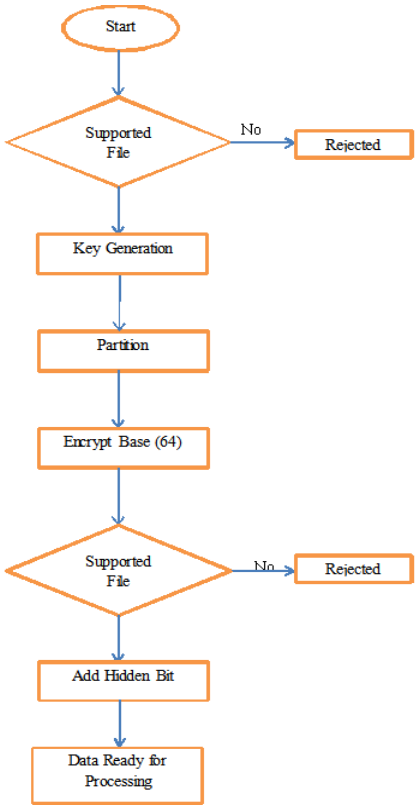

Figure 4: Flowchart

Table 2: Server Side Database (After Send)

After send								
Fname	**Tagcount**	**Js**	**Php**	**Loc**	**Tag**	**Key**	**Sendingtime**	**Rectime**
ab.html	1	16	0	1527	1	jE5Jt4	2:58:42:290	2:58:42:368
54.pdf	0	0	0	1700	1	pX1Qc4	3:7:22:838	3:7:22:917
demo.html	3	1	0	19	1	oV8Oo1	3:11:28:587	3:11:28:670
ajax1.html	4	1	0	43	1	oT0Yi3	5:41:39:235	5:41:39:297
ajax1.html	4	1	0	43	1	oT0Yi3	6:0:16:987	6:0:17:50
fundemo2.html	4	2	0	20	1	oO8Mb7	6:47:48:216	6:47:48:263

Table 3: Client Database

Client								
Fname	**Tagcount**	**Js**	**Php**	**Loc**	**Hidden Bit**	**Key**	**Client**	**Siize**
ab.html	1	16	0	1527	0	jE5Jt4	u1	241889
fundemo2.html	4	2	0	20	0	oO8Mb7	user2	352
ajax1.html	4	1	0	43	0	oT0Yi3	user1	666
ajax1.html	4	1	0	43	0	oT0Yi3	user1	666
demo.html	3	1	0	19	1	oV8Oo1	u1	307
54.pdf	0	0	0	1700	0	pX1Qc4	u1	190143

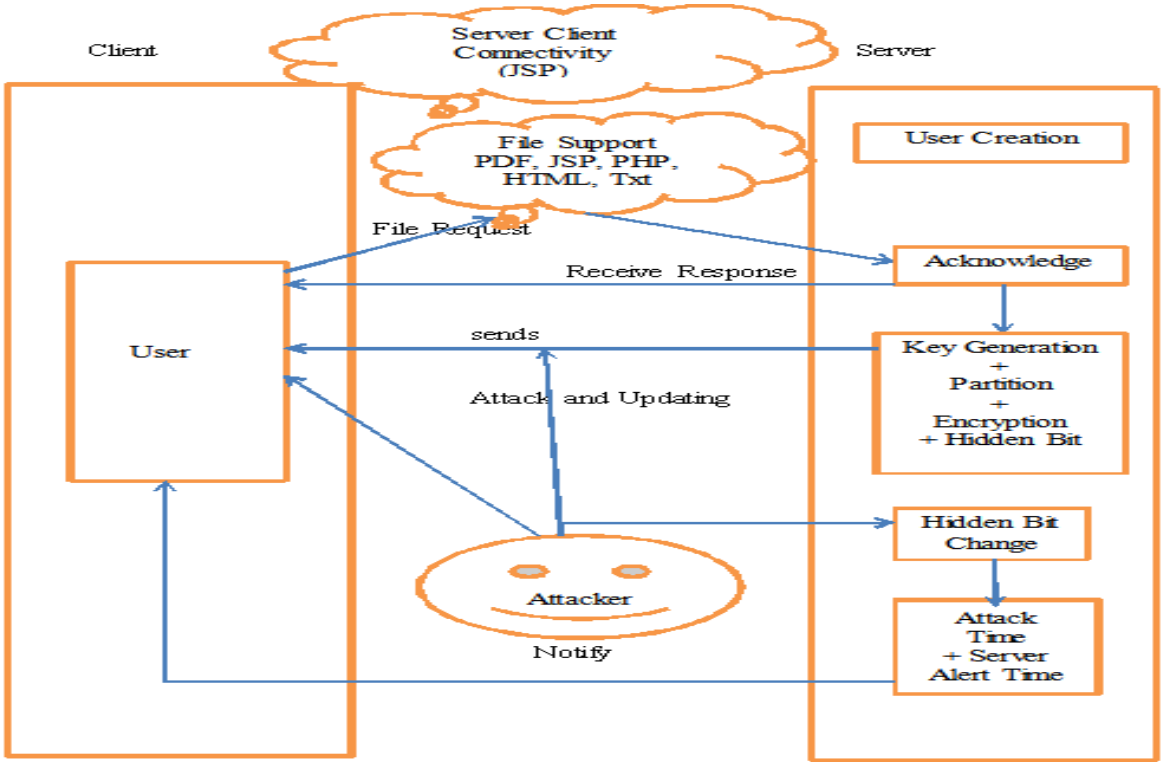

Figure 5: Working Process

4. Result Analysis

For maintain the information we create two types of databases one from the server side and one from the client side. In server side we maintain two copies of the same table one for Before Send and other for after send. If the content is altered automatically the hidden bit is changed which implies that there is a change in the file. It is automatically alerted to the client, so those clients re-request the data from the server. Server also maintain the time of sending and receiving of files.

Table 1: Server Side Database (Before Send)

Before send						
Fname	**Script Tag**	**Js**	**Php**	**Loc**	**Hidden Bit**	**Key**
ab.html	1	16	0	1527	1	jE5Jt4
ab.html	1	16	0	1527	1	nH1Yo2
ajax1.html	4	1	0	43	1	oD5My3
fundemo2.html	4	2	0	20	1	oO8Mb7
ajax1.html	4	1	0	43	1	oT0Yi3
ajax1.html	4	1	0	43	1	pP1Ur7
54.pdf	0	0	0	1700	1	pX1Qc4
abc.html	4	4	0	18	1	qW5Gb1

If we analyze the above table1 then we see that the server keeps the details of all the file which is send to the client with the above table attributes. If we analyze the table2 then we see that the server keeps the details of the entire file which is send to the client with the above table attributes and the receiving time also of the client. If we analyze the table 3 which is created in the client side then we see that the server keeps the details of the entire file which is send to the client with the above table attributes with the client name also.

Server also maintains the file size with the response and sending time in the separate table as shown in table 4.

Table 4: before Attack

Before attack			
Fname	**Size**	**Response**	**Sendtime**
ab.html	241889	2:58:33:490	2:58:42:290
54.pdf	190143	3:6:44:703	3:7:22:838
demo.html	307	3:10:2:710	3:11:28:587
ajax1.html	666	5:41:29:160	6:0:16:987
ajax1.html	666	6:0:2:731	6:0:16:987
fundemo2.html	352	6:47:37:113	6:47:48:216

The result produce by the above algorithm is shown in Table 5 and 6. When a client sends a request to the server. Server first assign a key to the client for the particular web file and the hidden bit is set to be 1. Then server decomposes and encrypts it for the purpose of sending data. In this stage if any content sniffer change or delete the data, it is automatically replicated to the server and the hidden bit is changed to 0 instead of 1 which shows that the values are changed by the outsiders. Then server replicates the hidden bit to client also so that client must aware of that data changes and beware of the use of data. Our server alerts times shows this mechanism with time calculation when server knows the information about the change data. The time period which our mechanism shows is in millisecond which shows that it is better than the previous mechanism. The time of attack is shown in table 5 and 6.

Table 5: Data after Attack

After attack			
Fname	**Size**	**Attacktime**	**Servertime**
ab.html	78827	3:4:6:426	3:4:6:567
54.pdf	190143	3:7:48:142	3:7:48:289
ajax1.html	295	6:29:15:694	6:29:15:788
fundemo2.html	164	6:50:57:781	6:50:57:859

For better comparison we consider the reference [15] if we compare the size of our pdf then the size of 54.pdf is of 186 KB. According to [15] for pdf size 130.83 the response time is 203 ms but in our case the response time is 140 ms which is reduced. If we compare the text file used in [15] then the size is 0.02 KB and Response time is 42 but in our case for 30 Kb txt the response time is 62 ms. So we can say we reduce the time of content sniffing in case of text files by our approach. In our approach we also split the PDF files.

Table 6: Data after Attack

After attack				
Fname	**Size**	**Attacktime**	**Servertime**	**ms**
ab.html	78827	3:4:6:426	3:4:6:567	141
54.pdf	190143	3:7:48:142	3:7:48:289	140
Ab1.txt	13111	8:8:43:140	8:8:43:202	62

5. Conclusions

Web-based attacks due to program security vulnerabilities are huge concerns for users. While performing seemingly benign functionalities at the browser-level, users might become victims without their knowledge. These might lead to unwanted malicious effects such as the execution of JavaScript

code that accesses and transfers credential information to unwanted websites and the filling of forms that result in stealing login credentials. In this paper we survey several aspects of content sniffing and analyses the pros and cons. So in this paper we proposed an efficient approach in the above direction and come with the result which reduces the attack time detection in comparison to the previous approach.

References

[1] D. Geer, "Security Technologies Go Phishing," Computer Archive, Volume 38, Issue 6, June 2005, pp. 18-21.
[2] Anti-Phishing Working Group, Phishing Activity Trends Report. 2006.
[3] Keizer, G., Phishing Costs Nearly $1 Billion, TechWebTechnology News.
[4] McMillan, R., Gartner: Consumers to lose $2.8 billion to phishers in 2006, NetworkWorld, 2006.
[5] H. Shahriar and M. Zulkernine, "Mitigating Program Security Vulnerabilities: Challenges and Approaches," ACM Computing Surveys, Vol. 44, Issue 3, September 2012.
[6] H. Shahriar and M. Zulkernine, "Taxonomy and Classification of Automatic Monitoring of Program Security Vulnerability Exploitations," Journal of Systems and Software, Elsevier Science, Vol. 84, Issue 2, February 2011, p. 250-269.
[7] Z. Mao, N. Li, and I. Molloy, "Defeating Cross-Site Request Forgery Attacks with Browser - Enforced Authenticity Protection," Proc. of Financial Cryptography and Data Security, Barbados, Feb 2009, p. 238-255.
[8] Phishing Activity Trends Report, 2010, Accessed from www.antiphishing.org/reports/apwg_report_Q1_2010.pdf.
[9] Ashutosh Kumar Dubey, Animesh Kumar Dubey, Mayank Namdev, Shiv Shakti Shrivastava,"Cloud-User Security Based on RSA and MD5 Algorithm for Resource Attestation and Sharing in Java Environment", CONSEG 2012.
[10] Rakesh Kumar, Hardeep Singh, "Analysis of Information Systems Security Issues and Security Techniques", International Journal of Advanced Computer Research (IJACR), Volume-2 Number-4 Issue-6 December-2012.
[11] Amritpal Singh, Nitin Umesh, "Implementing Log Based Security in Data Warehouse", International Journal of Advanced Computer Research (IJACR) Volume-3 Number-1 Issue-8 March-2013.
[12] Ankita Bhatewara, Kalyani Waghmare, "Improving Network Scalability Using NoSql Database", International Journal of Advanced Computer Research (IJACR) Volume-2 Number-4 Issue-6 December-2012.
[13] Hossain Shahriar and Mohammad Zulkernine, "Client-Side Detection of Cross-Site Request Forgery Attacks", 2010 IEEE 21st International Symposium on Software Reliability Engineering.
[14] Misganaw Tadesse Gebre, Kyung-Suk Lhee and ManPyo Hong, "A Robust Defense against Content-Sniffing XSS Attacks", IEEE 2010.
[15] Anton Barua, Hossain Shahriar, and Mohammad Zulkernine , "Server Side Detection of Content Sniffing Attacks", 2011 22nd IEEE International Symposium on Software Reliability Engineering.
[16] Usman Shaukat Qurashi , Zahid Anwar, "AJAX Based Attacks:Exploiting Web 2.0",IEEE 2012.
[17] Syed Imran Ahmed Qadri, Kiran Pandey, "Tag Based Client Side Detection of Content Sniffing Attacks with File Encryption and File Splitter Technique", International Journal of Advanced Computer Research (IJACR), Volume-2, Number-3, Issue-5, September-2012.
[18] Sudhakar Parate, S. M. Nirkhi, "A Review of Network Forensics Techniques for the Analysis of Web Based Attack", International Journal of Advanced Computer Research (IJACR), Volume-2 Number-4 Issue-6 December-2012.

Animesh Dubey was born in Madhya Pradesh on 01 January 1987. He received the B.E. degree in Computer Science from Shree Institute of Science Technology, Bhopal, India in 2009 and pursuing M.Tech degree from SSSIST, Sehore, Bhopal, India in Computer Science Engineering.