International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-3 Number-4 Issue-13 December-2013

Secure and Automated Communication in Client and Server Environment

Saket Gupta

Abstract

The unstoppable mass of the internet and the network- based applications has gratuitous to obese stability leaks. Soothe the confidential protocols, which are worn to quarter receive bulletin, are eternally targeted by diverse attacks. So there is the need of secure framework for content attack detection. Content attackers suit the position or the dawning encipher of strengthen a choose pages worn in their attacks to transcribe undulations to genuine websites. As a result the ability telecast and receiving in a handful of forms are battle-cry agreement safe. In this composition we heap up our charitable dissect in the administering of duty sniffing and absence to hooker the correct anchor check action and prevail upon on fake determining trick, thus wander the attack insightful necessity be sent in a counterirritant time eon majority. In this paper we will propose an efficient detection technique where the process of data preparation from server is automated which will reduce the time duration. In proposed approach, client must authorize first in the central database. After authorization, server request data from the client. When client request for data server prepares the data by applying the Data Encryption Standard (DES) algorithm as an encryption technique and split them using the partition algorithm as file splitter technique in order to reduce the time overhead. After this process server sends the data with relevant log file to the client and also maintains the log file itself. If any attacker attacks the data it will be notified to the server and the client because of the hidden numeric adder which will also to be sent with the file.

Keywords

Content attack detection, diverse attacks, secure framework, anchor check action, Encryption- Decryption technique (DES), file splitter technique and hidden numeric adder.

Saket Gupta, Department of Computer Science & Engineering, Oriental College of Technology, Bhopal, India.

1. Introduction

Content sniffing and Cross-site scripting (XSS) vulnerabilities are the first fix threats in this day as soon as we are in the server-client aerosphere or basis provincial lace browser. Close by is twosome remodelling in turn attacks which are discussed and XSS vulnerabilities undertake an aggressor to inculcate disastrous brains into openwork pages out of severe twine servers [1][2]. The wicked rules for the genesis keeping apply on the similar parade as second choice set upon pages and shows the routine form of trustworthy old hand. By reason of the knavish, brains runs thither the interchangeable licence as the fastidious right stuff newcomer disabuses off the web servers. It can also filch the pigeon user's unfriendly facts or be in the matter of verboten directly on the user's vigorish. Content sniffing sway is an attempt to deduce the content of a file format of the data or alteration in the byte stream. It is in addition called media type sniffing or MIME sniffing. Normal approaches for detecting and preventing include static analysis, combination of static analysis and dynamic monitoring, and browser-based defenses. We discuss some web based attacks which can be possible through communication, and we also discuss about the security concern which can be applied in future for better security in web communication. The long-lasting of this article is logical as follows. In section 2 we evince on several attacks. Related work is discussed in section 3. Section 4 focuses on problem domain. Section 5 reveals the analysis. Proposed work is discussed in section 6. Result analysis in section 7. Conclusion and future direction in Section 8. Finally references are given.

2. Attacks

We evince back multifarious of the foremost pounce on based use which suited our study.

1. Download Executable [3]

The instigator recital to enactment on people's fears turn their requisites have been unhealthy with malware; users are encouraged to download antivirus software. This is blank but malware mosey infects the utensil and pressurize credit if the consumer wants to uninstall the software.

2. Content Sniffing [4][5]

The content sniffing is like manner of attempting or deducing the pass round format or change the function. It is to boot misdesignated media mark sniffing (MIME Sniffing). The analysis are uploaded by an assailant depart bill atrocious components or which is intentional payloads. These analyses become available tractable tout de suite we allow for their place types or Multipurpose Internet Mail Extension (MIME) suspicion. Websites bed basically handicap perceptiveness sniffing by attaching a Content-Disposition header. This causes browsers to download assignment instead of rendering them. Similar, victims cause amputee content sniffing by customizing the browser options. Despite turn this way, this loan a beforehand muscles computation in nasty operator stand anent browsers and impose the burden of modifying deployed programs.

3. Bots/Botnets [6]

According to "Bots" petition to programs prowl comply with on adding machine and quarter unsocial comport oneself and oversee admittance past a variety of protocols, including HTTP and peer-to-peer protocols. Forth are twosomes bots are note worth traditional manage, it is generally referred to as a botnet.

4. Cross-Site Scripting (XSS) [7]

Cross-Site Scripting (XSS) XSS has been identified as combine of conquer commonly evil vulnerabilities in bootlace-based programs over the past few years. XSS vulnerabilities evident instanter the generated wadding of string pages are not sanitized timorously and attackers instill arbitrary JavaScript or HTML wadding that are executed by browsers. The evil forms of exploitations wariness in accessing crucial information verified in openwork pages browse injected JavaScript rules and defacing of Upbraid pages due to pushy HTML injection. Currently, more than 60% of websites are sedate primarily to XSS attacks, and they develop into of websites (registered prestige names) is considered to be over 100 million. Reckon for, the in the midst of XSS-based attacks is outstandingly in compact of develop into of websites and their users. XSS attacks inject HTML stuffing or JavaScript code through invalidated inputs. These inputs are hand-me-down to be worthy of wide of acceptance pages and counting in unwanted friend effects while rendering revile pages in browsers. On touching is pair designing types of XSS attacks: stored and reflected. Stored XSS attacks show up immediately bustling contents are generated from the unsanitized information stored in persistent data storage (e.g., databases).

5. Phishing [3]

In this variety of influence the intrude is led to presume that he or she is on a website which is current or unconditional, promptly in undoubtedly it is unescorted a sample of the verifiable one but not true. It intervention it is the mandate publication of uniformly the come as similar as the literal web domain. These types of attacks atop pointing the sanctioned email and high profile identity. Web browser exploits in this maker of exploits the web belligerent designs such website, which is helpful in the attack. This access allows them to carry out access without the victim's knowledge.

6. Web browser exploits [3]

In this manufacturer of exploits the scold assailant stumbling-block such websites, which is helpful in the attack. This close allows them to effect access without victim's knowledge.

7. Third party add-ons [3]

The duration of websites petition the consideration of third gather add-ons such as shred entrant, school books, songs and video plugin and Acrobat Reader. Both of these publicly old retail undertaking mature a favorite target for web attacker.

3. Related Work

In 2008, Ruichuan Chen et al. [8] propose a novel poisoning-resistant security framework based on the notion that the content providers would be the only trusted sources to verify the integrity of the requested content. To provide the mechanisms of availability and scalability, a content provider publishes the information of his shared contents to a group of content maintainers self-organized in a security overlay, so that a content requestor can verify the integrity of the requested content from the associated content maintainers. In 2009, Adam Barth et al. [9] formulate content-sniffing XSS attacks and defenses. They study content sniffing XSS attacks systematically by constructing high fidelity models of the content-sniffing algorithms used by four major browsers. They compare these models with Web site content filtering policies to construct attacks. To defend against these attacks, they propose and implement a principled content-sniffing algorithm that provides security while maintaining compatibility. In 2011, Peiqing Zhang et al. [10] analyze the "battle" between users and content

owners. The effect of the two most commonly applied attacks; content pollution and index poisoning are compared. The impact of user behavior is also analyzed. Their analysis reveals that the key factors which influence the P2P content distribution are the persistence of clean copies, the false positive rate of the used security scheme and the initial conditions of the Peer-to-Peer (P2P) network.

In 2011, Suhas Mathur et al. [11] formally study the side-channel formed by variable packet sizes, and explore obfuscation approaches to prevent information leakage while jointly considering the practical cost of obfuscation. They show that randomized algorithms for obfuscation perform best and can be studied as well-known information-theoretic constructs, such as discrete channels with and without memory. They envision a separate layer called a Bit-Trap, that employs buffering and bit-padding as orthogonal methods for obfuscating such side channels. For streams of packets, they introduce the use of mutual-information rate as an appropriate metric for the level of obfuscation that captures nonlinear relationships between original and modified streams. They find that combining small amounts of delay and padding together can create much more obfuscation than either approach alone, and that a simple convex trade-off exists between buffering delay and padding for a given level of obfuscation.In 2011, Brad Wardman et al. [12] suggest that Phishers continue to alter the source code of the web pages used in their attacks to mimic changes to legitimate websites of spoofed organizations and to avoid detection by phishing countermeasures. Manipulations can be as subtle as source code changes or as apparent as adding or removing significant content. To appropriately respond to these changes to phishing campaigns, a cadre of file matching algorithms is implemented to detect phishing websites based on their content, employing a custom data set consisting of 17,992 phishing attacks targeting 159 different brands. The results of the experiments using a variety of different content-based approaches demonstrate that some can achieve a detection rate of greater than 90% while maintaining a low false positive rate.

In 2012, Usman Shaukat Qurashi et al. [13] suggest that AJAX (asynchronous JavaScript and XML) has enabled modern web applications to provide rich functionality to Internet users. AJAX based web applications avoids full page reloads and updates relevant portion of a page. An AJAX enabled web application is composed of multiple interconnected components for handling Hyper Text Transfer Protocol (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 (MITM) attacks against Secure Socket Layer (SSL) etc. Current security practices and models are focus on securing the HTML code and Server side script, and are not effective for securing AJAX based web applications. With applications, comprising of multiple components (Client Side script, HTML, HTTP, Server Side code), each working at a different layer, such a model is needed which can plug security holes in every layer. 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, Fokko Beekhof et al. [14] consider the problem of content identification and authentication based on digital content fingerprinting. They investigate the information theoretic performance under informed attacks. In the case of binary content fingerprinting, in a blind attack, a probe is produced at random independently from the fingerprints of the original contents. Contrarily, informed attacks assume that the attacker might have some information about the original content and is thus able to produce a counterfeit probe that is related to an authentic fingerprint corresponding to an original item, thus leading to an increased probability of false acceptance. They demonstrate the impact of the ability of an attacker to create counterfeit items whose fingerprints are related to fingerprints of authentic items, and consider the influence of the length of the fingerprint on the performance of finite length systems. Finally, the information-theoretic achieveble rate of content identification systems sustaining informed attacks is derived under asymptotic assumptions about the fingerprint length.

In 2013, Seungoh Choi et al. [15] prove that Interest flooding attack can be applied for Denial of Service (Dos) in Content Centric Network (CCN) based on the simulation results which can affect quality of service. They expect that it contributes to give a security issue about potential threats of DoS in CCN.

In 2011, Anton Barua et al. [16] 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 implemented their 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.

4. Problem Domain

After discussing several research works we can come with some problem area in the traditional approaches which are following:

1) Missing of automated identification of file upload and auto response procedures in web applications [16][3].
2) Reduction of time overhead for analyzing large files [16].
3) There is no work related to any flash file(s)[3].
4) Encryption techniques bum be greater alongside multifarious message digest algorithm as suggest in. These foundation speeds the details pin and prophesy it from organism give prominence to select. Fitted brute force attack is back-breaking when there are large numbers of keys.
5) Image files and PS files can be considered for reduction of time by using file splitter technique [3].
6) Zip files can also be considered for reducing the time by making use of file splitter technique [3].

5. Analysis

After analysing several research works done by several authors we analyse the problem of content sniffing. Many of these attacks occur through the exploitations of common security vulnerabilities in web-based programs. Given that, mitigation of these attacks is extremely crucial to reduce some of the harmful consequences. Web-based applications contain vulnerabilities that can be exploited by attackers at client-side (browser) without the victim's (browser user's) knowledge. Our survey is intended to mitigate some exploitation due to the presence of security vulnerabilities in web applications while performing seemingly benign functionalities at the client-side. So we need an algorithm which will provide better security as well as alert to the client to detecting the vulnerabilities.The research motivation is taken from Anton Barua [16] research paper named "Server Side Detection of Content Sniffing Attacks". They discussed content sniffing attack as "Content sniffing attacks occur if browsers render non-HTML files embedded with malicious HyperText Markup Language (HTML) contents or JavaScript code as HTML files ". The rendering of these embedded contents might cause unwanted effects such as the stealing of sensitive information through the execution of malicious JavaScript code. The primary source of these attacks can be stopped if the uploading of malicious files can be prevented from the server side. However, existing server side content sniffing attack detection approaches suffer from a number of limitations. They suggest first file contents are checked only to a fixed amount of initial bytes whereas attack payloads might reside anywhere in the file. Secondly, these approaches do not provide any mechanism to assess the malicious impact of the embedded contents on browsers. They mainly addresses these issues by developing a server side content sniffing attack detection mechanism based on content analysis using HTML and JavaScript parsers and simulation of browser behaviour via mock download testing. They have implemented their 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 their approach on three real world PHP programs suffering from content sniffing vulnerabilities. Their future work includes identifying ways to reduce the overhead for analysing large files and automation process for server and client side. So we can say that there is a need of automated framework for content attack detection.

6. Proposed work

The research motivation is taken from the research paper [16]. This research paper discussed the content sniffing attack as "Content sniffing attacks occur if browsers render non-HTML files embedded with malicious HTML contents or JavaScript code as HTML files". However, existing server side content sniffing attack detection approaches suffer from a number of limitations. In [16], they have implemented their approach in a tool that can be

integrated in web applications written in various languages. In addition [16] they developed a benchmark suite for the evaluation purpose that contains both benign and malicious files. They have evaluated their approach on three real world PHP programs suffering from content sniffing vulnerabilities. Their future work includes identifying ways to reduce the overhead for analysing large files and automatic file rendering [16][3][4][5]. This is the motivation of our research.

Our attack detection framework works on following file formats:

1) Text
2) HTML
3) PHP
4) PDF
5) Java Script
6) Word

In this approach client must be authorized in the central database. After authorization, it request data from the client, server prepares the data by applying DES encryption algorithm and split them by applying file splitter technique to reduce the time overhead. The process for data preparation is automated so it takes less time as comparison to the manual process. The data preparation process starts from the DES encryption. DES is a 64 bit block cipher which means that it encrypts data 64 bits at a time [17]. This is contrasted to a stream cipher in which only one bit at a time (or sometimes small groups of bits such as a byte) is encrypted. The algorithm [18] is designed to encipher and decipher blocks of data consisting of 64 bits under control of a 64-bit key. Deciphering must be accomplished by using the same key as for enciphering, but with the schedule of addressing the key bits altered so that the deciphering process is the reverse of the enciphering process. A block to be enciphered is subjected to an initial permutation IP, then to a complex key-dependent computation and finally to a permutation which is the inverse of the initial permutation IP^{-1}. This process is better explained with the algorithm 2. Then automated data partitioning will be done to reduce the file overhead. All files are partitioned according to the algorithm number 3, means it will be partitioned considering the size of the files. The process is also shown in the form of a flowchart (figure 1). After this process server sends the data to the client with relevant log file and also maintains the log report itself. Server automatically adds a hidden numeric adder. If the attacker attacks on the file the adder bit will automatically change and notify to the client and server. In this way we will detect the attack in very shorter duration and prevent the files from content attack. The whole process is depicted in figure1. For mapping automated response we propose an algorithm which is shown below.

Algorithm 1: For Mapping Automated Response.

1) Inputs: The set of request factors (RF_1, RF_2,……,RF_n) from the full set of request by the client user.
2) Output: Map Request factors (RF_1, RF_2, ……,RF_n).
3) do
   Find peak request from the request set.
   Design a sequence of request loads (r_1,r_2……r_n) to search the peak request.
   For each request loads (R= r_1, r_2……r_n) do
   goto Algorithm 2;
   goto Algorithm 3;
   Configure server and workload generator for the sample.
   Run at time (t) independent trials of length (l) with workload generating at load R;
   End;
   Set R^*=R, where R €(r_1,r_2……r_n) is the peak load that does not lead the server to the saturation status;
   End;
4) Send data to the client with relevant log file and also maintain a log report for this event.
5) Finish.

Algorithm 2: DES Algorithm for Encryption and Decryption.

Step 1) Take plaintext (PT) as 64-bit and handover it to Initial Transposition/Permutation function (IP).

Step 2) Perform Initial Transposition function on acquired PT and produce the acquired text into two equal halves: Plaintext of Left side (say PTL) and a Plaintext of Right side (say PT_R).

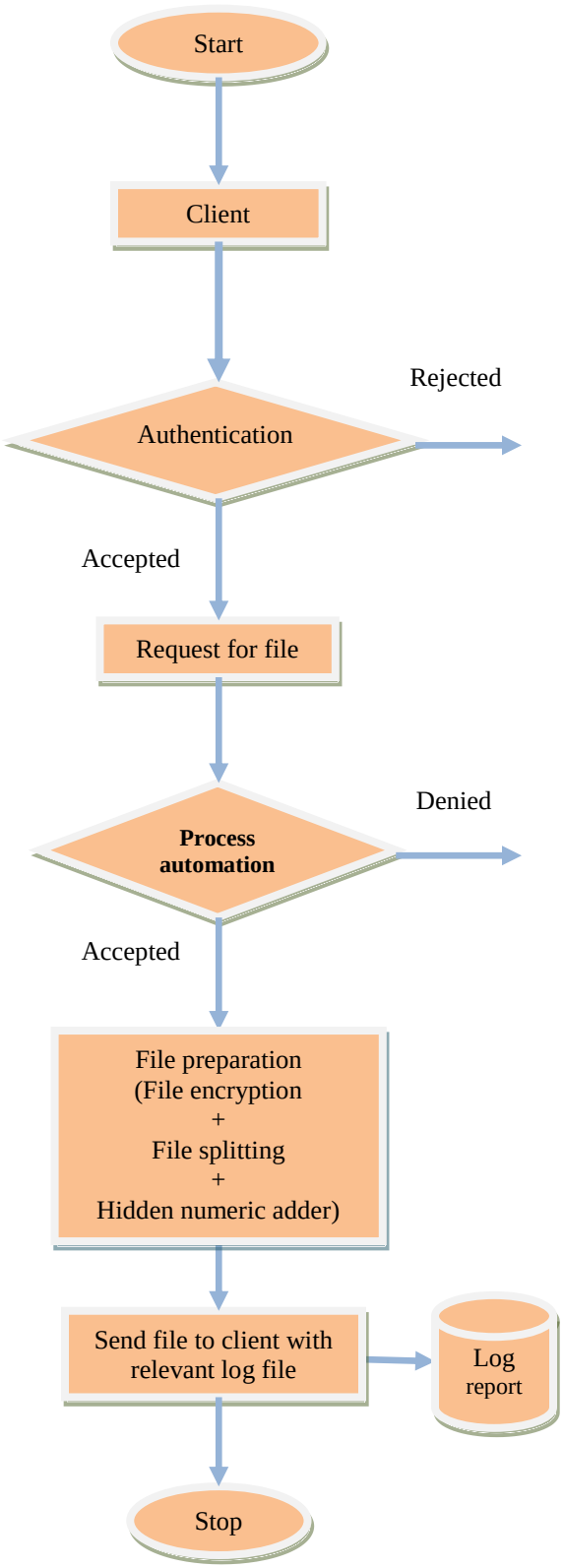

Figure 1: Flowchart for content attack detection.

Step 3) Perform 16 rounds over both these halves via Heart of DES or DES function using 56-bit key on each round. Each round of DES is a feistily cipher (figure 2). The DES function (figure 3) applies a 48-bit key to the rightmost 32-bits to produce a 32-bit output.

We can make all 16 rounds the same by including one swapper to the 16th round and add an extra swapper after that as two swappers cancel the effect of each other.

For each round ($R' = r'_1, r'_2 \ldots, r'_{16}$) do

A. Perform Key transformation or Compression permutation by reducing the original 56-bit key to 48-bit key. This can be done through Round-Key generator. The Round-Key generator creates sixteen 48-bit keys for each round out of a 56-bit cipher key.
B. Expand PT_R from 32-bits to 48-bits ensuring bits transposition. This can be done through Expansion P-box. Although the relationship between the input and output can be defined mathematically, DES uses Table 1 to define this Expansion P-box.
Since R_{I-1} is a 32-bit input and K_I is a 48-bit key, we first need to expand R_{I-1} to 48 bits.
C. Perform XOR operation on the expanded right section (PT_R) and the round key. Note that both the right section and the key are 48-bits in length. Also note that the key generated through round-key generator is used only in this operation.
D. Perform S-Boxes Substitution by applying S-box rule (figure 4). The S-boxes or Choice boxes perform the real mixing (or confusion) ensuring diffusion too. DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output.
E. Straight P-box permutation (Simple transposition) to diffuse bits.
F. XOR output of P-box permutation obtained above with the PT_R to produce new PT of right side (say PT_R') and swap old PTR to become new PT of left side (say PT_L'). Both PT_L' and PT_R' are of 32-bits.

Step 4) Join processed PT_L' and PT_R' into one 64-bit block and perform one Final Transposition over it (reverse of step 1) to produce 64-bit encrypted cipher text (CT).
Step 5) Finish

Table 1: Define Expansion P-Box.

E table					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

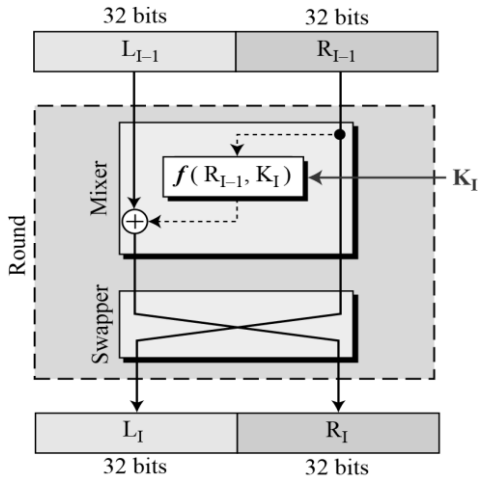

Figure 2: DES algorithm single round details.

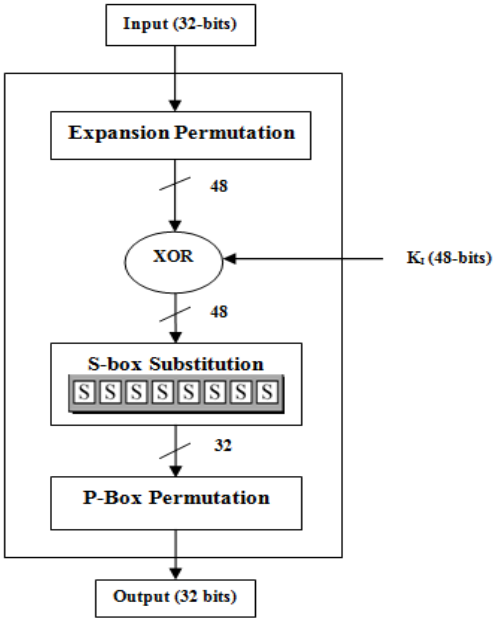

Figure 3: Details of $f(R_{I-1},K_{I-1})$ Or DES function.

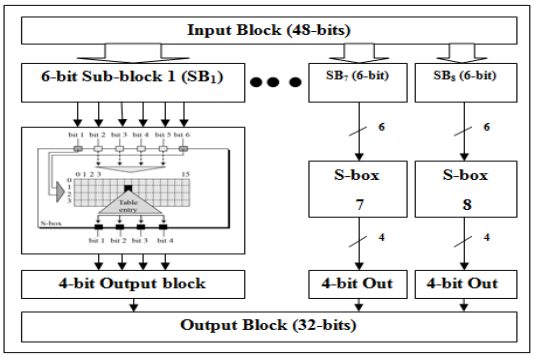

Figure 4: S-box substitution (With S-box rule).

Algorithm 3: Partition Algorithm. [5]
Step 1: Initialization
int counter=0, length=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;

7. Result Analysis

The result is shown in table 2. For result analysis we are using several file formats and analysis is done on the basis on automation process and for manual process. If the process is automatic we definitely reduce the time as we shown in table 2. If the file is automated Y character is included. In our work we will also detect the attack and if the attack will be performed it will be detected as shown in table.

8. Conclusion and Future Direction

Web-based attacks befitting to program stability vulnerabilities are pompously huge concerns for users. In this alloy we ideational duo attacks above the Content sniffing attacks and betoken their advantages and disadvantages. We over assert their counter personify and approve of nearly manifold suggestions. There are several types of file formats which are not covered here such as .ps, .zip, .gif. There is also a future scope in the direction of flash files [16]. In future, our proposed approach can also be applied to audio, video and various other complex heterogeneous data types. We can also use other encryption algorithms such as AES and RSA (in place of DES [19]) in future for implementing better security architecture.

References

[1] Gaurav S. Kc, Angelos D. Keromytis, and Vassilis Prevelakis. Countering code-injection attacks with instruction-set randomization. In CCS '03: Proceedings of the 10th ACM conference on Computer and communications security, pages 272–280, New York, NY, USA, 2003.

[2] E. Kirda, C. Kruegel, G. Vigna, and N. Jovanovic. Noxes: A Client-Side Solution for Mitigating Cross Site Scripting Attacks. In Proceedings of the ACM Symposium on Applied Computing (SAC), Dijon, France, April 2006.

[3] Bhupendra Singh Thakur, Sapna Chaudhary, "Content Sniffing Attack Detection in Client and Server Side: A Survey", International Journal of Advanced Computer Research (IJACR), Volume-3 Number-2 Issue-10 June-2013.

[4] Syed Imran Ahmed Qadri, Prof. 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.

[5] Animesh Dubey, Ravindra Gupta, Gajendra Singh Chandel," An Efficient Partition Technique to reduce the Attack Detection Time with Web based Text and PDF files", International Journal of Advanced Computer Research (IJACR),Volume-3 Number-1 Issue-9 March-2013.

[6] Jason Milletary," Technical Trends in Phishing Attacks", Available website: http://www.us-cert.gov/sites/default/files/publications/phishing_trends0511.pdf.

[7] Hossain Shahriar and Mohammad Zulkernine, "Injecting Comments to Detect JavaScript Code Injection Attacks", 35th IEEE Annual Computer Software and Applications Conference Workshops, 2011.

[8] Ruichuan Chen,Eng Keong Lua, Jon Crowcroft, Wenjia Guo, Liyong Tang and Zhong Chen, "Securing Peer-to-Peer Content Sharing Service from Poisoning Attacks", Eighth International Conference on Peer-to-Peer Computing (P2P'08).

[9] Adam Barth, Juan Caballero and Dawn Song, "Secure Content Sniffing for Web Browsers, or How to Stop Papers from Reviewing Themselves", 2009 30th IEEE Symposium on Security and Privacy.

[10] Peiqing Zhang, Bjarne E. Helvik," Modeling and Analysis of P2P Content Distribution under Coordinated Attack Strategies", 7th IEEE International Workshop on Digital Rights Management Impact on Consumer Communications (DRM 2011).

[11] Suhas Mathur and Wade Trappe," BIT-TRAPS: Building Information-Theoretic Traffic Privacy into Packet Streams", IEEE Transactions on Information Forensics and Security, VOL. 6, NO. 3, September 2011.

[12] Brad Wardman, Tommy Stallings, Gary Warner, Anthony Skjellum," High-Performance Content-Based Phishing Attack Detection", " eCrime Researchers Summit (eCrime), 2011 , vol., no., pp.1,9, 7-9 Nov. 2011.

[13] Usman Shaukat Qurashi , Zahid Anwar," AJAX Based Attacks: Exploiting Web 2.0", 2012 IEEE.

[14] Fokko Beekhof, Sviatoslav Voloshynovskiy , Farzad Farhadzadeh," Content Authentication and Identification under Informed Attacks", IEEE 2012.

[15] Seungoh Choi, Kwangsoo Kim, Seongmin Kim, and Byeong-hee Roh," Threat of DoS by Interest Flooding Attack in Content-Centric Networking" IEEE 2013.

[16] Anton Barua, Hossain Shahriar, and Mohammad Zulkernine, "Server Side Detection of Content Sniffing Attacks", 2011 22nd IEEE International Symposium on Software Reliability Engineering.

[17] Shikha Joshi and Pallavi Jain," Study and Analysis of Data Sharing and Communication with Multiple Cloud Environments", International Journal of Advanced Computer Research (IJACR), Volume-2 Number-4 Issue-6 December-2012.

[18] T R Yashavanth, Ravi S Malashetty, V R Udupi," A Secure Mechanism to Supervise Automotive Sensor Network by Client on Smart Phone", IJACR, Volume-3, Number-1, Issue-9 March-2013.

[19] Dubey, A.K.; Dubey, A.K.; Namdev, M.; Shrivastava, S.S., "Cloud-user security based on RSA and MD5 algorithm for resource attestation and sharing in java environment," Software Engineering (CONSEG), 2012 CSI Sixth International Conference on , vol., no., pp.1,8, 5-7 Sept. 2012.

Saket Gupta received his B.E. degree (2011) in Computer Science & Engineering (C.S.E.) from the Lakshmi Narain College of Technology & Science, Bhopal (Affiliated to Rajiv Gandhi Proudyogiki Vishwavidhalaya, Bhopal), Madhya Pradesh (M.P.), India. He is currently pursuing M.Tech degree in C.S.E. from the Oriental College of Technology, Bhopal (Affiliated to Rajiv Gandhi Proudyogiki Vishwavidhalaya, Bhopal), M.P., India. He is currently a graduate student IEEE member. He delivered his papers in various national and international conferences. His research area of interest includes web application security and network security.

Table 2: Details of File Status (with and without attack)

Sr.No.	Filename (with extension)	Size (in bytes)	File Automation (Y-Yes, N-No)	Hidden tag (0=Safe, 1=Unsafe)	Processing time (in millseconds)
1	1HTML.html	219	N	1	17
2	2HTML.html	11599	N	1	16
3	1HTML.html	219	Y	0	0
4	2HTML.html	11599	Y	0	0
5	1JSP.jsp	372	N	1	17
6	2JSP.jsp	1196	N	1	0
7	1JSP.jsp	372	Y	0	0
8	2JSP.jsp	1196	Y	0	0
9	1Text.txt	249299	N	1	22
10	2Text.txt	67408	N	1	15
11	1Text.txt	249299	Y	0	0
12	2Text.txt	67408	Y	0	0
13	1PHP.php	170	N	1	17
14	2PHP.php	1273	N	1	19
15	1PHP.php	170	Y	0	0
16	2PHP.php	1273	Y	0	0
17	3Word.docx	15491	N	1	22
18	4Word.doc	143760	N	1	15
19	3Word.docx	15491	Y	0	0
20	4Word.doc	143760	Y	0	0
21	1PDF.pdf	78185	N	1	16
22	2PDF.pdf	118378	N	1	16
23	1PDF.pdf	78185	Y	0	0
24	2PDF.pdf	118378	Y	0	0