Client Side Detection of MIME Sniffing

Siddhesh Dabholkar[1], Rakesh Patil[2], Uttam Vaishnav[3]

Jyothi Arun[4], *Assistant Professor*

Department of Information Technology, Atharva College of Engineering, Mumbai University
Maharashtra, India

[1]s.dabholkar99@gmail.com,[2]rakeshpatil5192@gmail.com,[3]uttamvaishnav3@gmail.com,
[4]amritajyothi87@gmail.com

***Abstract* — Due to the increase in web technology most of the people rely on the Internet for their daily routine. People access data without knowing the vulnerabilities in it. Nowadays without security measures data might be vulnerable to an attack. These attacks can either get access to monitor the information, or can modify the information which can eradicate the data. MIME (Multipurpose Internet Mail Extensions) sniffing vulnerabilities are the major security threats today when we are in the server-client environment or using any web browser. It 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. We implement private key cryptography to encrypt the requested data by the client from the server. We use file splitting technique to minimize file processing time. This helps the File execution to take place within milliseconds. To notify any modification in the data we are implementing Tag bit concept. We use message digest algorithm to generate tag bits. .**

***Keywords*— MIME sniffing, cryptography, tag bit, MD5 algorithm.**

I. INTRODUCTION

Today, most of us use internet to fulfil most of our daily routine activities. This may include social networking, emails, online money transactions and much more. As this internet communications includes sensitive information like passwords, account number or other secret data we want to ensure its security. With growth in the field of technology it is now possible to intercept the online communication and retrieve the confidential data. This attack allows the attacker to fetch or modify the data without making the user aware of such modifications. So in order to prevent such attacks, it is necessary to implement security mechanisms to ensure the safety of the sensitive information on the Internet.

After studying various analyses on network security we can conclude that there is much vulnerability still present in the transmission procedure when user accesses the web. MIME sniffing attacks is one of the threats that take place between such communications. MIME sniffing attacks occur when a browser treats non-HTML files as HTML files. These non-HTML files may be injected with malicious code which will be executed when the user opens it. For example if the attacker manipulates the content in a way to be accepted by the web app and rendered as HTML by the browser, it is possible to inject code in an e.g. image file and make the victim execute it by viewing said image.
When we are preventing the attack from the server side and it will be notified to the client then we can prevent the attack.

II. PROBLEM DEFINITION

A web browser handles all the files or data that the user request from the web server. The type of the file is considered by the web browser in order to handle the requested file. For an instance, the browser will handle image files differently than text files. Mostly the web server will specify correct type of file through a Content-Type header, while few web servers specify incorrect Content-Type header. This feature of the web browser to determine the correct content type of the file is called as MIME sniffing. This is also known as Content sniffing. This feature has a step that checks the first 256 bytes of a file against a list of defined headers. This feature helps the user to successfully browse the web, but can also turn into a source of attack. Many security issues have been discussed in the past, with its prime focus on web applications that permits users to upload images. The MIME sniffing algorithm checks only the content-type header of the images. if a web application allows user to upload images and check only its file extension, the user might upload an image.jpg file which contains malicious HTML code. This could lead to an attack.

There are several attack detection approaches that are deployed at program runtime [7][8][9][10][11].Various detection mechanisms for MIME sniffing attacks have been developed till date. However these mechanisms have limitations of their own. We have identified limitations among few mechanisms which are:

1) Many mechanisms will assume that all web based programs are trusted and authorized.
2) Most of them depend on the alteration of client and server environments or the type of communication that takes place between these sides.
3) Many approaches do not completely address to these network attacks such as MIME sniffing, SQL injection and many more.

4) These attack detection approaches often takes too long especially in case of large files.

III. RELATED WORK

In 2009, Adam Barth et al. formulate content-sniffing XSS attacks and defences. 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.

In 2011, Anton Barua et al. Developing a server side content sniffing attack detection mechanism based on content analysis using HTML and JavaScript parsers and simulation of browser behaviour via test downloads. They implemented their concept using a tool which can be integrated in web applications written in various languages. They also developed a proper framework for evaluation purpose that contains both benign and malicious files.

In 2012, Syed Imran Ahmed Qadri et al. Provide a security framework for server and client side. 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.

In 2013, Animesh Dubey et al. propose an efficient partition technique for web based files (jsp, html, php), text (word, text files) and PDF files. They are working in the direction of attack time detection. For this motivation they 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. They also apply partitioning method on PDF files. There result comparison with the traditional technique shows the effectiveness of their approach.

IV. PROPOSED APPROACH

In this paper we provide a security mechanism on the server side as well as on the client side. This approach will detect MIME sniffing attack on the server side and alert it on the client side.

A. Working Process

The working process of the mechanism is shown in figure 1.

When client wants to get data from the server side it will first establish a secure connection to the server. Client requests for the data from the server and the admin will provide the required data if it is available in the server database. The server will provide the data by first encrypting it by using the concept of private key cryptography. It uses the same key for encryption and decryption process, which is also known as symmetric key cryptography.

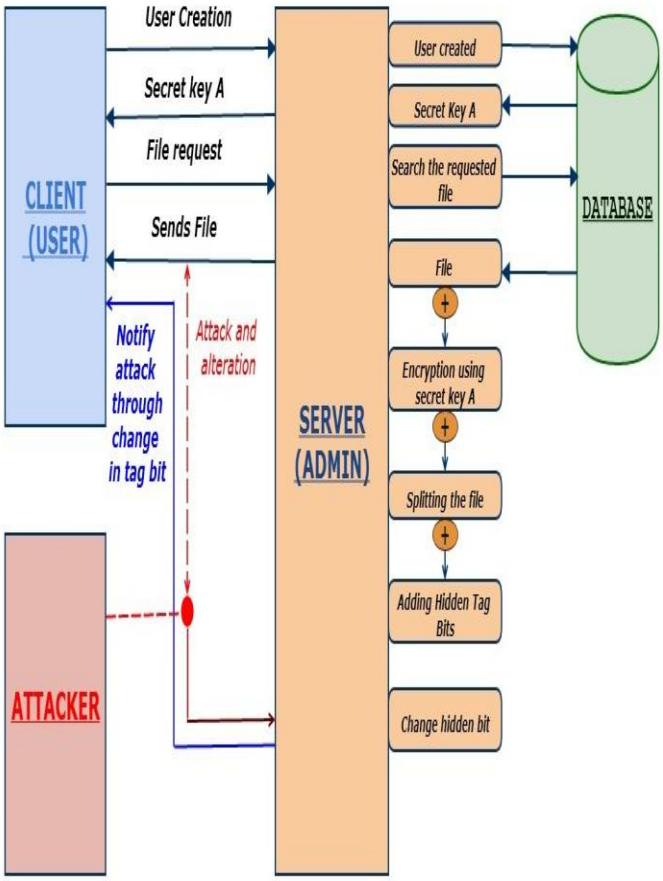

Fig 1: Working Diagram

To send encrypted messages to each other every user should be in the possession of the same key. The same key will be used by the sender to encrypt their message before sending it to the receiver. At the receiver end the same key will be used by the receiver to decrypt the message.

The end users should protect the secret key properly, if not could lead to an attack. If an attacker is able to intercept this secret key he will be able to read all the encrypted messages that are being transmitted between the users. Therefore protection of this secret key is very important. Base64 encode and decode is used in java defined by RFC 2045 that provides symmetric key encryption.

Conceptually, symmetric is simple which is a secret decoder ring model. Encryption and Decryption are done using the same secret decoder ring. Take an example of a door lock, which can be locked and unlocked using the same door key.

After a successful encryption process comes the splitting part. The file will then be spitted according to the length. This will reduce the complexity and time span of sending the file to the user. At the end we also add the hidden tag bit to the file. The purpose of the tag bit is to notify the client that an attack has occurred and alteration of data has taken place. A memory buffer is also included which will detect the content alteration. Message Digest algorithm is used for this part.

As shown in Figure 1, if an attacker tries to intercept the communication and modify the data, tag bit will change and this attack will be notified to the client.
Two types of databases are created for managing the information. One database is from the server side and another database is from the client side. At the server side, we manage two copies of the same table for before send and after send. These tables includes the username, filename, tag count, tag bit and the secret key that will be used for encryption and decryption process. When the content in the file is automatically modified the tag bit will be 1 which will indicate that the contents in the file have been altered. The client will be informed about this alteration so that those clients would re-request data from the server.

TABLE I
File info before send

Username	File name	Tag count	Tag bit	Key
rani	page.html	813	0	6xygmk
rani	web_sample1.html	433	0	Zimmrf

TABLE III
Client access info

File name	Tag count	Tag bit	Key	Send time	Receive time
page.html	813	1	6xygmk	13:49:8:59	13:49:8:9
web_sample1.html	433	1	Zimmrf	13:56:11:6	13:56:11:71
html-hell.html	332	1	kIZAJq	14:0:56:6	14:0:56:56

TABLE IIIII
File info after send to client

User name	File name	Tag count	Tag bit	Key
rani	page.html	813	1	6xygmk
rani	web_sample1.html	433	1	Zimmrf
rani	html-hell.html	332	1	kIZAJq

V. ALGORITHM

A. Message Digest 5 (MD5) Algorithm

Message Digest (MD5) algorithm was developed in 1991 by Ronald L. Rivest. The message digest 5 (MD5) algorithm takes as input a message of any arbitrary length and produces a 128-bit message hash as output. In MD5 terminology message hash is also called as message digest. Of the input message while the MD5 algorithm is computationally complex to understand, a simple description is provided in this section. The MD5 algorithm is defined as a sequence of the following steps.

1) Append Padding Bits
The input message is padded or extended such that its length (in bits) modulo 512 is 448. In other words the message is extended in a manner so as to make it just 64 bit less than being the multiple of 512 bits.

2) *Append length*
Next, a 64 bit representation of the length of the message is appended to the end of message. The length of the message is the length before padding bits where added. In case the length is greater than 2^64 than, only the lower order 64 bit of the representation of the length (in bits) are used. After appending the length to the message, the resulting message now as a length that is the exact multiple of 512 bits.

3) *Divide into blocks*
After the message is made a multiple of 512 bits, it is divided into blocks of 512 bits each. Each block of 512 bit is itself considered a sequence of sixteen 32 bits word. The mD5 algorithm is then applied on each of these blocks in a sequence, as described in the next step.

4) *Apply MD5 algorithm on each block.*
The MD5 algorithm is block chained hashing algorithm. In MD5 a hash function is first applied to the 1st block in the message and initial seed value is provided as a parameter to the hash value, along with the 512 bit message block. The output (hash) of the first block is then added with the seed itself and the sum then becomes the seed for the next seed itself, and this sum then becomes the seed for the next block. When the hash of the last block is completed its value becomes the hash for the entire message. Thus the seed for each block depends on the hash of the previous block and hence block cannot be hashed in parallel, but hashed in sequential fashion. At the heart of MD5 algorithm are 4 auxiliary functions each of which take has input three 32 bit words and produce as output one 32 bit word. This auxiliary function is used to perform the hashing for each of the block in the entire message.

5) *Output the hash.*
Message digest produce for the entire message is the output of the hash function when it is applied to the last block in the message. This hash is a 128 bit sequence, and works as message digest for the entire

message. The strength of the MD5 algorithm arise from the fact it is computational infeasible to produce two message having the same message digest, or to produce any message having a pre defined message digest. Since the MD5 algorithm is designed to work with 32 bit words, it is quite fast when run on 32 bit machine.

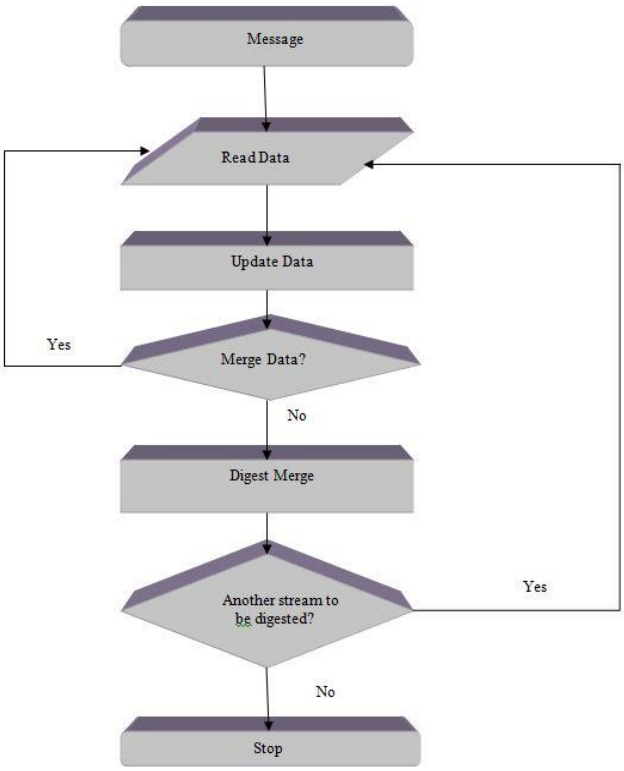

Fig 2: Flow Chart

VI. CONCLUSION AND FUTURE SCOPE

Attacks on the internet are taking place due to the network security problems, User might become victim of various vulnerabilities of web without alerting them. This can lead to stealing of sensitive information .In this paper we provide security mechanism for both client and server Side. In Future our work for content sniffing attack detection includes ways to reduce the overhead for large files and to evaluate our approach for some other file types. We convert the content type of any file into HTML and Plan to find an automated way to do so. The future work will also include the Automatic detection of file uploads procedures to Integrate our filter.

A. References

[1] 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

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

[3] 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

[4] Animesh Dubey, Ravindra Gupta, Gajendra Singh Chandel,‖ An Efficient Partition Technique to reduce the Attack Detection Time with Web based

[5] Misganaw Tadesse Gebre, Kyung-Suk Lhee and ManPyo Hong, ―A Robust Defence Against Content- Sniffing XSS Attacks‖, IEEE 2010.

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

[7] Y. Zhang, J. Hong, and L. Cranor, ―CANTINA: A Content-based Approach Detecting Phishing Websites ,Proc. of the 16th International Conference on World Wide Web (WWW),Banff, Alberta, Canada, May 2007.

[8] M. Alalfi, J. Cordy, and T. Dean, ―WAFA: Fine-grained Dynamic Analysis of Web Applications, Proc. of the 11th International Symposium on Web Systems Evolution (WSE), Edmonton, Canada, Sept 2009, pp. 41-50 .

[9] M. Gundy and H. Chen, ―Nonce spaces: Using Randomization to Enforce Information Flow Tracking and Thwart Cross-site Scripting Attacks,‖ Proc. of the 16th Annual Network and Distributed System Security Symposium (NDSS), San Diego, California, USA, February 2009.

[10] Y. Nadji, P. Saxena, and D. Song, ―Document Structure Integrity: A Robust Basis for Cross site Scripting Defence, Proc. of the 16th Annual Network and Distributed System Security Symposium (NDSS), San Diego, California, USA, February 2009.

[11] T. Jim, N. Swamy, and M. Hicks, ―Defeating Script Injection Attacks with Browser- Enforced Embedded Policies,‖ Proc. of the 16th International Conference on World Wide Web, Banff, Alberta, Canada, May 2007, pp.601-610.

