Detection of Javascript Vulnerability At Client Agen

Saurabh Jain, Deepak Singh Tomar, Divya Rishi Sahu

Abstract- These days, most of companies expanding their business horizon through dynamic web sites based on Web 2.0 concept. The JavaScript is a key choice of web developers to build sophisticated dynamic web 2.0 application such social network site, blogs, e-commerce websites. On the other hand vulnerable JavaScript code is also exploited by the hackers to launch the attacks. Hacker may tamper the JavaScript code to perform attacks against the client's browser. In this work the series of attacks such as click-jacking, password capturing, phishing and cookies stealing are developed to understand to affect of vulnerable JavaScript code on web browser. The detection of vulnerable JavaScript code is a tedious task for security experts. Hence signature and regular expression based matching mechanism has been developed to detect the vulnerable JavaScript code.

Index Terms - Web Security, JavaScript, Web Browser, JavaScript Interpreter, Malware, Phishing, Clickjacking.

1 INTRODUCTION

In web environment the scripting attacks through JavaScript are common security threat. These threats are exploit sensitive data, authorization schemes, defraud users, and defame web sites. The attacker launches these attacks to leak information, stealing password or loading malware to the victim's system through vulnerable JavaScript code. Client-side JavaScript statements which are embedded in HTML tags can be processed by web browser [2]. On request of client/browser, the server responds to the whole content of the document, along with all HTML and JavaScript statements. The browser reads the page, visually displaying the HTML results and executing JavaScript statements as it goes. Traditionally JavaScript has runs in the web browser, but now the attention in passing it to the server side as well. JavaScript is different from other scripting languages because the developer uses JavaScript to take full control over the web browser, performing simple computations on the client side, validating the user's input and generating HTML code on-the-fly.

This paper is organized as follows:

The section I deal with the introduction of the paper, section II describes the browser components, section III describes challenges, section IV explores the attacks through JavaScript at client agent with attack scenario, and section 5 discusses detection technique and section 6 presents future work and concludes the paper.

1.1 Web Application Vulnerability

Web-based applications have become highly sophisticated of vulnerabilities which are capable of increasing exploiting rate.

- *Saurabh Jain,Maulana Azad National Institute of Technology,Bhopal, India*
  Email- data.saurabhjain@gmail.com
- *Deepak Singh Tomar, Maulana Azad National Institute of Technology Bhopal, India,*
  Email-deepaktomar@manit.ac.in
- *Divya Rishi Sahu, Maulana Azad National Institute of Technology,Bhopal, India*
  Email-divyarishi.sahu@manit.ac.in

2 BROWSER COMPONENT IN RESPECT OF JAVASCRIPT EXECUTION

Web browser is application based software which is used for accessing, presenting, and negotiating information resources on the World Wide [1]. The architecture of a web browser is shown in Figure 1. It contains eight major subsystems. The functionalities of browser's components are as follows:

- **The User Interface**–Provide an interface for comfortable navigation to end user.
- **The Browser Engin**e – Supply the actions between the user interface and the rendering engine.
- **The Rendering Engine** - The main job of rendering engine is parsing. It also display of the requested contents on the browser panel. By default display HTML images and documents, displaying PDF using a PDF viewer plug-in.
- **Networking**- It is responsible for network calls, like HTTP requests. It has platform independent API that handles multiple layers of the networking model (Network, Session and Presentation layers).
- **JavaScript Interpreter-** The JavaScript interpreter used to parse and execute the code.It evaluates JavaScript code, which may be embedded in web pages. Scripts to be parsed and executed immediately when the parser reaches a <script> tag. The parsing of the document halts until the script was executed.
- **UI Backend-** UI backend provide drawing of essential widgets like windows and combo boxes.
- **Data Persistence-** Data Persistence allows to keep up web page information, state, variables and styles. It also preserves information in the browser's history cookies and session files to store information.

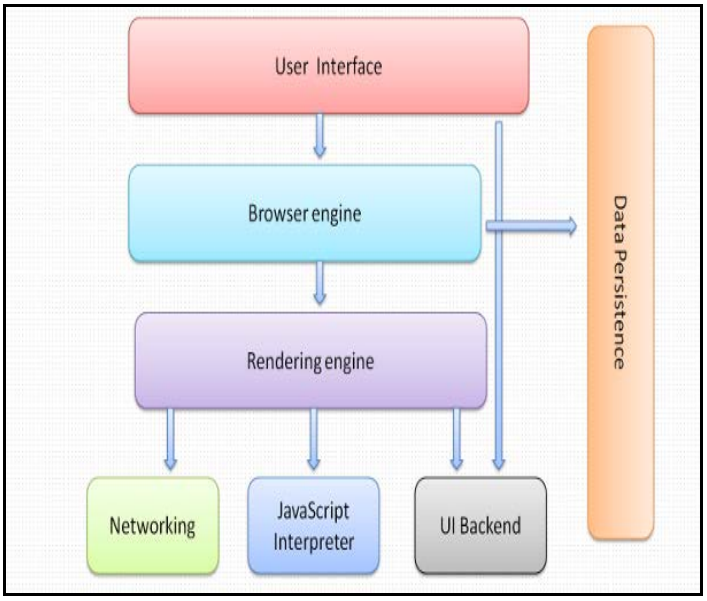

Figure 1 Architecture of Web Browser

2.1 Script Execution on Web Browser

The following are the common step of script execution through browser:

a) The browser loads the html.
b) The browser starts to load the external resources from top to bottom line by line.
c) If a <script> is found, the loading of html will be blocked and wait till the JavaScript file is loaded and executed and then continue.
d) Other resources (CSS/Image) are loaded in simultaneously and executed if needed.

2.2 Client Side Attacks through Browser

Client side attack exploits the vulnerability in client application running on client agent browser such as Mozilla firebox, chrome, internet explorer etc.The following are the attacks that may be exploited by hacker through browser.

2.2.1 Attacks against Plug-ins within the Web browser

Plug-in is a set of software components that adds specific abilities, customize the functionality of the browser, like play video, display new file types [4]. It also changes the browser settings, block the client request and download malware to the user's system.

2.2.2 Clickjacking

Clickjacking is a malicious technique that can force the user to adjusting the user's system settings without the user awareness [5]. By using JavaScript, an attacker could place a fake button or link under or over a genuine button or link.

2.2.3 Phishing

The Phishing is attempting to acquire information such as usernames, passwords, and confidential details by impersonating as a reliable web page [6].

2.2.4 Cookie Stealing

Cookies are raw data which is sent by the server and stored on client system for later retrieval to allow user-side customization of web information. The information in the cookies is easily accessible by attacker, with the aid of these cookies the attackers can steal sessions [7], and compromise accounts.

2.2.5 Stealing Information from Storage

The browser cache and browser history [8] are also precious portions of information that attackers can gain access.

3 CHALLENGES

JavaScript is one of the most simple, flexible and effective languages used to extend functionality of websites. There are a number of challenges which are faced within the development in JavaScript. It also possesses negative effects that reflect on implementation JavaScript. One of the major aspects is data security, privacy, and data leak and confidentiality loss. When JavaScript snippets appended onto web pages that execute on browser, exploit malicious effects on client system. There are few browser engines that are render JavaScript differently and provides results in inconsistent functionality.

3.1 Browser Manipulation through JavaScript

JavaScript manipulates attributes of the browser, control the size of the client window, the URL which it's requesting, the appearance of its toolbars, and many other properties like deleting or changing the page form and browser history.

4 POSSIBLE ATTACKS THROUGH JAVA SCRIPT

4.1 Clickjacking

A clickjacked page tricks a user into performing unwanted actions by clicking on a hidden link or button. In these pages the attackers load another page in a transparent layer (Hidden iframe) through JavaScript [5]. The users think that they are clicking visible buttons, while they are actually performing actions on the hidden page.

Possible Attacker: An attacker having good knowledge of CSS and Iframe and explore his knowledge to generate hidden page in background layer.

Vulnerability: The websites that contained embedded malicious JavaScript code to take illigal can take action without user's awareness.

Impact of attack: Revealing confidential information from their system while clicking on harmful link or button.

The code shown in Figure 2 that dynamically generates a transparent web page:

```
<script>
   <iframe
        id="new"
src=http://localhost:8084/Attacks/DANGER%20PAGE.html
         name="iframe_a"
         style = opacity: 0;
         position:absolute;
        top:313px;left:212px;width:500px;height: 200px;">
   </iframe>
</script>
<form action= "DANGER PAGE.html"
    method="post"
    name="form"
class="style">
<input type="submit"
    name="btn"
    id="btn"
    value="Don't Click">
</form>
```

Figure 2. Code snippet vulnerable to clickjacking attack

Attack Scenario : Initially an Attacker send malicious link to victim via email or other means which contains a hidden iframe, to perform malicious task. The innocent victim provides credentials and clicks the login button shown in figure 3, Login button is directly above the fake button but fake button is invisible. Victim wouldn't be able to see it, but victim would still be able to click on it. After clicking on it causes the real malicious affects shown in figure 4, inside the iframe, it may be downloading a virus to the victim's system ,fund transfer through victim's Bank account, it may be click on an advertise and other sophisticated attack may be performed. The innocent victim will be unaware of click.

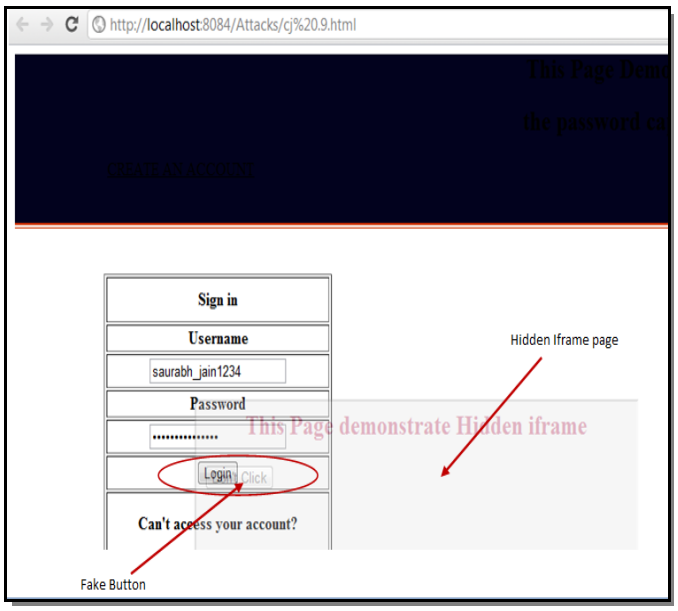

Figure 3. Hidden iframe - containing fake page and the fake button

Figure 4. Activated hidden iframe

4.2 Capture Password through JavaScript

Modern browsers provide the functionality of remembering user id's and passwords. By use of this option, browser automatically saves user's id and passwords in the browser's memory. As web browsers logically hide passwords using unreadable form like asterisks or dots. A JavaScript code used to acquire passwords in web browsers along with an alert box.

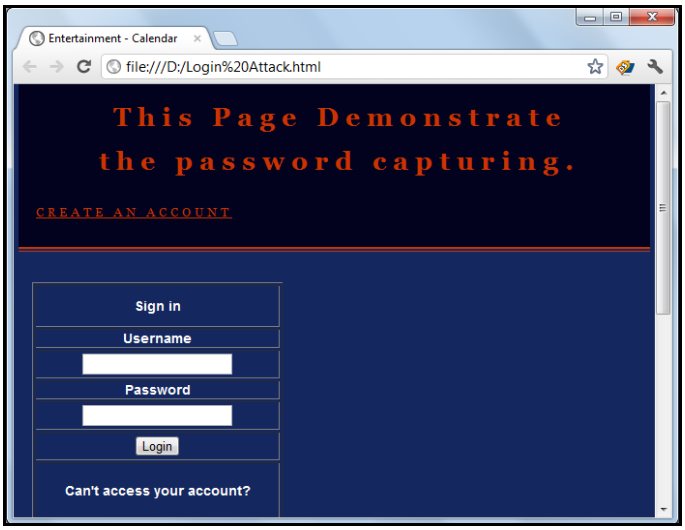

Figure 5. Login page

Possible attacker: An attacker having knowledge of tricky JavaScript code to exploit the system.

Vulnerability: Victim chooses or click remembers username and passwords option those stores in browser.

Impact of Attack: Loss of confidential and authentic information.

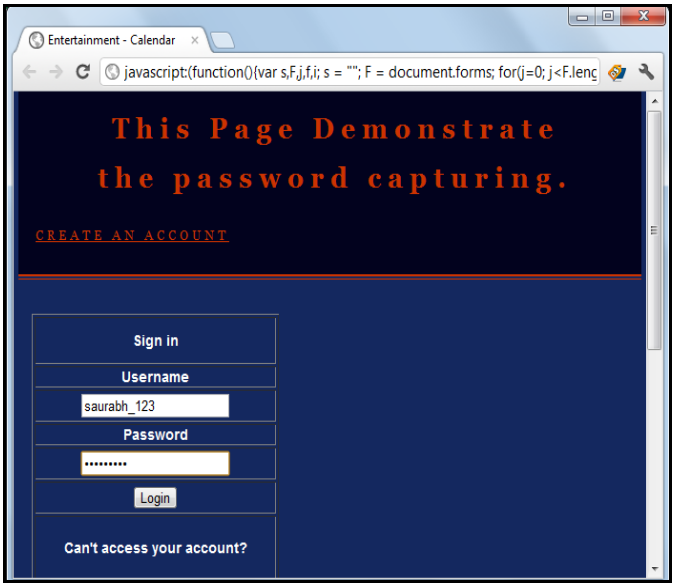

Figure 6. Login page with credentials

Attack Scenario: When user provide credentials to login to a particular website as shown in the figure 5, after clicking on login button a popup window shows a message "To remember username and password". If user selects this option remember the username and password. The attacker can get access to password by executing JavaScript code as shown in address bar of the browser that shown in figure 6. The attacker can retrieve victim's password along with an alert box as shown in the figure 7.

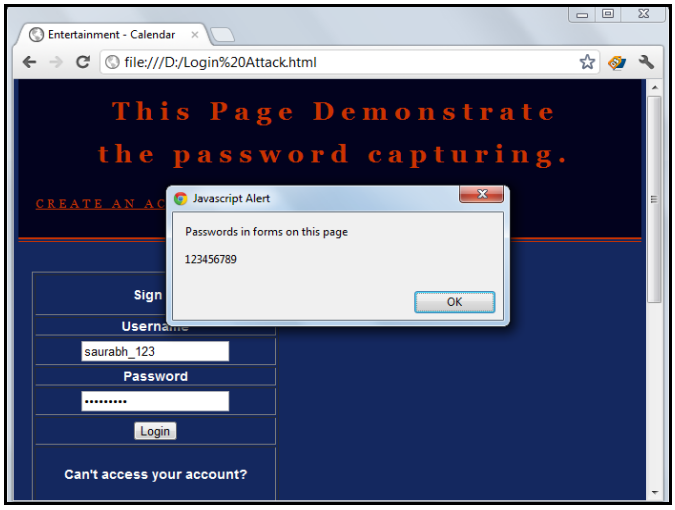

Figure 7. Alert box along with password

4.3 Phishing

Phishing exploits users to submit their credentials and passwords to websites by forging those sites and convincing the user that the site is authentic [7].

Possible attacker: An attacker trapping a user to click on a link, via having link or an email under attacker control.

Vulnerability: This attack involves setting up a clone website that appears identical to the website with which the victim has a relationship, such as a social networking site.
Impact of Attack: Loss of victim's personal or private information, loss of money, gaining access to sensitive data.

4.4 Cookie Stealing

The attacker can steal cookies inserting malicious JavaScript code into a web page of a vulnerable website. This script collects user cookies and then sends them to the attacker.

```
<SCRIPT> document. write(document. cookie)
```

Figure 8. Code snippet of cookie stealing

Possible attacker: An attacker having knowledge to insert malicious JavaScript code into vulnerable website, such as in a forum post, profile or blogs.

Vulnerability: Innocent user click on unknown or unwanted page or link.

Impact of attack: The attacker has now stolen the session and can impersonate the user, hijack an account.

Phishing and Cookie stealing attacks are practical experimented but the snapshots of these attacks are not included due to limited size of paper.

5 Detection Technique

This section presents a malicious JavaScript code detection technique. A number of detection techniques (e.g., static analysis, testing and scanning are widely used before program deployments to detect injected code.

5.1 Proposed Work:

In this section a technique is proposed to detect vulnerable JavaScript code which is based on signature and regular expression matching. In this work the extension of firebug is explored to trace the JavaScript code on web browser.

5.2 Signature based Detection (static rules)

The signature based detection defines static rules [3] which have to be defined before the analysis made.

5.2.1 Hidden Iframe

The <iframe> tag specifies an inline frame. An inline frame is used to embed another document within the current HTML document. The CSS property for transparency is opacity. Attacker embedded iframe with opacity in code that can execute without the user's knowledge and perform malicious task. Based on this concept rule 1 is generated.
Rule 1: If iframe is used with opacity 0 (CSS property). Then (the code is said to be vulnerable).

5.2.2 Number of dynamic code executions

The number of function calls that are used to dynamically interpret JavaScript code (eval and setTimeout) and the number of DOM [10] changes that may lead to executions

(document. create Element). Based on this concept rule 2 is generated.

Rule 2: If the dynamic interpretation of calling eval, setTimeout and document.createElement functions. Then (the code is said to be vulnerable).

5.3 Rule Based Algorithm

These rules are applied in the algorithm proposed shown in figure.

```
INPUT: HTML Web page source code assigned in S

OUTPUT: Alert if malicious code detected.

Extract all JavaScript codes in S and assign in W.

Search malicious code

String S, W;

int Count;

 If (Found iframe  with opacity 0 in W)
       Alert: Code is vulnerable
End

Else If (Count (eval ||setTimeout|| document. Create
      Element)>1 in W)
      Alert: Code is vulnerable
End

Else
       Alert: No vulnerability found
End
```

Figure 8 Rule Based Algorithm

5.4 Regular Expression (Regex)

A regular expression (regex) is a text string which search for a pattern. Regular expressions facilitate a flexible and efficient text processing [9]. The goal of a regular expression is to match a certain expression within a chunk of text. Regular Expression Library provides a searchable database of regular expressions. A regular expression pattern is usually enclosed within slashes ('/').

Example: /[A-Za-z0-9] /

5.5 Evaluation

This section present detection of the malicious script by applying signature based rules and generates the regular expression which is shown in figure 10 and its explanation shown in Table 1.

Malicious code shown in figure 9 performs clickjacking attack on web browser.

```
<script>
   <iframe src
="http://localhost:8084/Attacks/DANGER%
     20PAGE.html"   opacity:0;
   </iframe>
</script>
```

Figure 9.Code snippet of click jacking attack

The following regular expression recognizes tags:

```
/([<script>]+)([<iframe  src=]+)(https?|ftp|file|http)://[a-zA-Z0-
9+&@#/%?=~_|!:,.;]*+([opacity:0;\\s]+)([\\/<iframe>\\s]+)([
\\/<script>]+)/
```

Figure 10. Regular expression to detect clickjacking attack

Table 1 Explanation of Regular Expression

Regular Expression	Purpose
[<script>]	Check <script>
[<iframe src=]	Check <iframe src=
(https|ftp|file|http)://	Check the protocols
[A-Za-z0-9]	Check alphanumeric values
[&@#/%?=~_|!:,.;]	Check special characters
\s	Check whitespace character
[opacity:0;]	Check opacity:0
[\\/<iframe>]	Check </iframe>
[\\/<script>]	Check </script>
*	Use for string occurs zero or more times
+	Use for checking one or more character
( )	Use for grouping the regular expression

6 Conclusion and Future Work

In this paper, attacks such as clickjacking, password capturing, phishing and cookies stealing are explored with attack scenarios to understand the affect of vulnerable JavaScript code on web browser. During the research work it is found that mostly these attacks are imposed through JavaScript code. Hence a novel method has been developed and proposed to detect the vulnerable JavaScript code at client agent, research also generate regular expression to speed up the process of vulnerable JavaScript detection and regular expression based approach is enforced. The future work includes implementation of more efficient detection technique with more sophisticated signature. The future work also evaluate runtime overhead of proposed approach.

Acknowledgment

The research presented in this paper would not have been possible without our college, at MANIT, Bhopal. The Authors wish to express our gratitude to all the people who helped turn the World-Wide Web into the useful and popular distributed hypertext it is, and also wish to thank the anonymous reviewers for their insightful feedback.

Reference

[1] Alan Grosskurth, Michael W. Godfrey, "Architecture and evolution ofthe modem web

browser", http://grosskurth.ca/papers/browser-archevol-20060619.pdf

[2] D. Flanagan. JavaScript: The Definitive Guide, 4th Edition. December 2001.

[3] G. Wassermann and Z. Su, “Static detection of cross-site scripting vulnerabilities”, ICSE, Germany, 2008, pp. 171-180.

[4] Plig-ins http://www.boutell.com/newfaq/definitions/plugin.html

[5] Franco Callegati and Marco Ramilli University of Bologna “Attack Trends - Frightened by Links”

[6] Huajun Huang; Shaohong Zhong; Junshan Tan “Browser-Side Countermeasures for Deceptive Phishing Attack” Information Assurance and Security, 2009. IAS '09. Fifth International Conference**.**

[7] Chuan Yue; Mengjun Xie; Haining Wang “Automatic Cookie Usage Setting with CookiePicker” Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on 25-28 June 2007.

[8] Nadeem, T.; Killam, B. ”A study of three browser history mechanisms for Web navigation” Information Visualisation, 2001. Proceedings. Fifth International Conference

[9] Regular expression http://docs.oracle.com/javase/1.4.2/docs/api/java/util/regex/Pattern.html

[10] Feng Zhao “The Algorithm Analyses and Design about the Subjective Test Online Basing on the DOM Tree” Computer Science and Software Engineering, 2008 International Conference