Plug-in integration to a web browser for vulnerability analysis in JavaScript

Rafael Enrique Rodriguez-Rodriguez 1, Daniel Camilo Escobar-Ravelo2
Luis Felipe Herrera-Quintero3, Jaime Chavarriaga4
1, 2, 3, 4 Computer Science Engineering
Catholic University of Colombia
Bogotá D.C., Colombia
{rerodriguez55, dcescobar99, lfherrera, jachavarriaga}@ucatolica.edu.co

Abstract- Nowadays the information security is one of the priorities for big, medium and small sized companies, for that reason techniques and tools have been developed to improve the security within programming language, avoiding the spread of vulnerability called BadSmells within application codes and websites. The novelty of this paper focuses on design and implementing a Plug-in that is integrated to the web browser and takes advantage of JavaScript code analyzer JsPrime scanner, all this in order to avoid the insertion of malicious scripts and safety drawbacks.

Keywords: BadSmell, Plug-in, JavaScript, JsPrime

I. INTRODUCTION

Computer security is critical to the operations carried out through internet. The information security is a key component for organizations and individuals, providing reliability and certainty of safe operations. Today there are a variety of techniques and tools to improve safety; however, there are problems to solve as security on JavaScript [1], focusing primarily on vulnerabilities JavaScript code.

Frequent cases of attacks on web browsers (Chrome, FireFox, Internet Explorer, etc.) are often based on the use of JavaScript code. Which is an interpreted programming language that provides easy access to information that is running in a browser; Therefore, Internet users may experience data leakage such as passwords or banking information compromising privacy and security.

Mentioning the problem in which the subject's BadSmells [2] develops, tend to be mitigated by scanners code [3], which allow visualizing and analyzing results on different vulnerabilities that can be exploited for future attacks.

Prior to 2005, the vast majority of security experts and developers paid little attention to BadSmells [17], in this context, JavaScript was not a point of attention, so users still surfing internet without knowing the risks. In that same year the first major XSS worm, managed to shut down the popular social net-working Web site MySpace. The payload being relatively benign, the Samy Worm was designed to spread from a single MySpace user profile page to another, finally infecting more than a million users in only 24 hours[17]. Other cases of attacks are presented over time and this triggered many attacks affecting the major websites, for this reason XSS arguably stands as the most potentially devastating vulnerability facing information security and business online [17].

In light of this, security companies began developing methods of prevention, creating filters to detect attacks. Esprima a high performance, standard-ECMAScript compliant parser written in ECMAScript, and it was implemented in JsPrime. This tool can interpret malicious JavaScript code and tell the user what kind of threat is [18]. For this reason, JsPrime scanner and implementing a Plugin, it gives the opportunity for a common user to identify easily BadSmells.

Therefore, this article describes the integration of a Plug-in is proposed to analyze a web browser BadSmells and avoid inserting malicious scripts and disadvantages of security.

The remainder of this paper is organized as follows. Section II describes the contextualization of the work. Section III proposed Plug-in architecture. Section IV focuses on the analysis of the results obtained by the implementation of the Plug-in. Finally, Section V presents the conclusions of the work done.

II. CONTEXTUALIZATION AND RELATED WORK

In much of the world, a change in technology and the continuous evolution of the Internet has made these become the planet and in the process, information security has become more important for the development of large numbers of processes larger network.

In this sense, every day faces various security issues, which mean lots of vulnerabilities. Therefore the development of the internet has also designed a large number of web browsers, which allow the user to interact with the network of internet networks.

Have developed various programming languages that support browsers and one that has been successful is JavaScript [4], however the activation of this language has meant that the web acquire dynamism, but in turn takes lot of vulnerabilities.

Also, JavaScript has become a global language, its variables have general scope and functions are global, for this reason has been developing tools and mechanisms to detect and counter security attacks based browser [5], [6].

Moreover, the security project open source web applications (OWASP) released the list of most frequent attacks [7], which can be analyzed by IronWasp and JsPrime.

In the world, we can see that there is a constant approach to the web browsers improvement on the internet security...
issue, and one of those improvements is to promote and encourage everything related with JavaScript, actually there are projects that are in charge if avoiding the malicious code filtration. Therefore, to create a Plug-in containing an scanner that can identify code irregularities symptoms, also known as BadSmell, in order to apply it in a web browser that has the JavaScript console active, promoting its security.

III. PROPOSAL

A. Description

IT security in internet is determinant, and JavaScript is one of the most used languages for giving dynamism, also nowadays is one of the biggest preoccupations of developers, due to the security gaps and code vulnerabilities that it presents. When JavaScript interpreted language is ran, it allows the malicious actors to give scripts through internet and run them in user’s computers. That means it can generate passwords leaks, cookies theft, etc.

To contribute to the safety of users on the Internet, we developed a Plug-in integrated JsPrime; the extension is designed to be integrated into Google Chrome, JavaScript vulnerable to analyze code [9]. For this, when the extension is activated JavaScript code of the web page browsing, and then this code is used by JsPrime eventually led to the results of the analysis made on the page and are displayed is extracted by user by providing information of potential risks.

By making the analysis of the tools like JsPrime as the scanner of vulnerable code and Google Chrome as the Web browser, we looked for alternatives for the programming language we will use in the development of the Plug-in and its integration to the browser.

The decision is to use JavaScript to develop the Plug-in, due to JsPrime is entirely developed in this language; the analysis of vulnerabilities is of JavaScript.

To carry out the development of Plug-in is used as JsPrime scanner and integration thereof to a Plug-in for Google Chrome, considering the use of the Google API developer for creating the [10].

B. Plug-in Architecture

The architecture of the Plug-in is local, which means that the execution of the program takes place entirely in a single server. It consists of four modules as shown in Figure 1. A module of Google Chrome that represents the presence of the web browser, the form on the web page that may contain HTML, CSS, PHP and / or JavaScript within which it can be a Badsmells, the Plug-in module containing the code scanner JsPrime.

The system architecture is composed by four subsystems, the first is based on the extraction of the scripts of the website and provides a user interface, the second is the connection that has the first subsystem analysis and reporting, the third is analysis subsystem that is responsible for analyzing the JavaScript code, the fourth print the report with the test results (see figure 2).

Likewise, the user interacts with the browser, which in turn displays the contents of a website where this has code to be scanned by the Plug-in, and also the user interacts in implementing the Plug-in to analyze code.
And therefore derived from the architecture of the Plug-in is shown in Figure 3 the interaction for the user to browse a certain website and use the Plug-in to analyze the code of the desired page.

In the previous figure can be seen that the user accesses the Internet via a PC where the user installed the Google Chrome web browser, which provides access to a certain web page containing the user thinks that there are risks and use the Plug-in to scan the JavaScript code website, JsPrime internally analyzes the code and save the results so that the user can display a tab with vulnerabilities having the website.

C. Plug-in Development

Google Chrome has a way of developer [11]; allowing anybody to program over this platform; Plug-ins and extensions as this browser call them.

To make the development it necessary to have files of the JsPrime code scanner [12], attached to two necessary files for integrating the Plug-in to the browser, which are manifest that allows giving permissions, including an user interface and the icon file that is an image used as an user interface with a 19 x19 size [13].

Also in the manifest file is programed a function to run emergent pages, which is going to be the result tab, also the browser action code is written, which is a function that indicates where the interface button of the Plug-in has to go, general information like: name, Plug-in version and we page to run, when we click in the interface.

Chrome offers two alternatives for the location of the interface button [14]. Also must say that the works are to principal JavaScript vulnerabilities in HTML, and having the script code in a web page JavaScript be analyzed.

Finally, the Plug-in should be charged to the browser and the user interface button in the upper right corner of the browser was displayed (see Figure 4). Then enter any web page and just click the button to the Plug-in and you can analyze JavaScript code in web pages, for this reason the following jobs to be met by the Plug-in arise:

- Extract Script from the actual web page
- Analyze web pages that have scripts made with JavaScript language.

D. USE OF THE PLUG-IN

The user must access the web browser, when installed the Plug-in, entering any web page and simply click on the button interface Plug-in so that later you can see a tab with the results of the analysis as shown in Figure 5 and Figure 6.

![Figure 4: Plug-in](image)

![Figure 5: Execute Plug-in](image)

![Figure 6: Results tab](image)
IV. TEST AND RESULTS

We can see proves of a better performance, using the statistic tool of JsPrime and the implementation of the Plug-in, with the approach of validating the analyzing time and how long would the user take to analyze this code without availability of the Plug-in. To develop the test, we will take into account 5 cases of BadSmell proves, provided by JsPrime and Esprima [15], which are going to be inserted in a HTLM file and then analyzed through the extension, all this in the case of the Plug-in, and in the case go the web tool, the scripts will be copied in a manual way.

<table>
<thead>
<tr>
<th>Description</th>
<th>Time (sec) Static</th>
<th>Time (sec) Plug-in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Print the page numeral symbols</td>
<td>6.45</td>
<td>1.12</td>
</tr>
<tr>
<td>Replaces numerals symbols of a web page</td>
<td>7.3</td>
<td>2</td>
</tr>
<tr>
<td>Create a &quot;div&quot; tag where it says the pound sign if you meet the conditions</td>
<td>6.14</td>
<td>1.4</td>
</tr>
<tr>
<td>Puts the identified anchor tag with the redir Variable</td>
<td>5.76</td>
<td>1.11</td>
</tr>
<tr>
<td>Associates two variables and param node in to an array</td>
<td>7.05</td>
<td>1.26</td>
</tr>
</tbody>
</table>

**Table 1:** Codes for analysis

Below are the results of time in graphs; the reduction of the required time for the analysis is evident [16], with the implementation of JsPrime tool in the Plug-in.

![Figure 7: Performance Test](image)

Can be demonstrated in Figure 7 which time the scan tool into the Plug-in is minimal compared to the tool without Plug-in.

Likewise, the tool was tested on websites, so obtaining results of any threats that the user may experience when browsing, this information can be seen in table 3.

<table>
<thead>
<tr>
<th>link of the page</th>
<th>state</th>
<th>Type vulnerability</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://submanga.com/Naruto/677/223182">http://submanga.com/Naruto/677/223182</a></td>
<td>detected</td>
<td>NON-ACTIVE SOURCE</td>
<td>1</td>
</tr>
<tr>
<td><a href="http://www.kongregate.com/games/R2Games/wartune">http://www.kongregate.com/games/R2Games/wartune</a></td>
<td>detected</td>
<td>NON-ACTIVE SOURCE</td>
<td>4</td>
</tr>
<tr>
<td><a href="http://www.ucatolia.edu.co/paw/">http://www.ucatolia.edu.co/paw/</a></td>
<td>detected</td>
<td>NON-ACTIVE SOURCE</td>
<td>1</td>
</tr>
<tr>
<td><a href="http://www.yaske.to/">http://www.yaske.to/</a></td>
<td>detected</td>
<td>NON-ACTIVE SOURCE</td>
<td>1</td>
</tr>
<tr>
<td><a href="https://www.mintransporte.gov.co">https://www.mintransporte.gov.co</a></td>
<td>detected</td>
<td>Active Sink</td>
<td>1</td>
</tr>
<tr>
<td><a href="http://espansol.free-ebooks.net/">http://espansol.free-ebooks.net/</a></td>
<td>Not detected</td>
<td>Ninguno</td>
<td>0</td>
</tr>
<tr>
<td><a href="https://developer.chrome.com/home">https://developer.chrome.com/home</a></td>
<td>detected</td>
<td>NON-ACTIVE SOURCE</td>
<td>1</td>
</tr>
<tr>
<td><a href="http://www.youtube.com/">http://www.youtube.com/</a></td>
<td>Not detected</td>
<td>Ninguno</td>
<td>0</td>
</tr>
<tr>
<td><a href="http://www.rojadireccta.me/">http://www.rojadireccta.me/</a></td>
<td>Not detected</td>
<td>Ninguno</td>
<td>0</td>
</tr>
<tr>
<td><a href="http://www.fichajes">http://www.fichajes</a></td>
<td>Not detected</td>
<td>Ninguno</td>
<td>0</td>
</tr>
</tbody>
</table>

It be evidenced the decrease of time in Figure 8 for the Plug-in code analysis vulnerable JavaScript.
V. CONCLUSIONS

- Alternative security for Internet users was presented. The implementation of an extension or plug-in to review the content of any webpage and give results they notify the user that the page that this is vulnerable or not, to navigate safely, to that effect, allows a person with no computer skills can be quiet while surfing the internet.

- It was possible to attach the scanner JsPrime code to the web browser via a plug-in, so take advantage of the characteristics of the scanner, allowing the analysis of codes contained in the web pages.

- It could be concluded that the integration of the browser plug-in to analyze using JsPrime vulnerable code, minimizing the analysis time compared to static analysis tool.

ACKNOWLEDGEMENTS

We thank the Catholic University of Colombia, because of its contribution in the development and the experiments

VI. REFERENCES


[7]. OWASP Board Member, Unraveling some of the Mysteries around DOM-based XSS: OWASP Top 10 Project Lead [online]

Table 2: Web pages analysis


[18]. JavaScript static security analysis made easy with JSPrime, Nishant Das Patnik & Sarathi Sahoo