A Google Chromium Browser Extension for Detecting XSS attack in HTML5 based Websites

Arun Prasath Sivanesan
Department of Electrical Engineering and Computer Science
University of Toledo, Toledo, OH, USA
arunprasath.sivanesan@utoledo.edu

Akshay Mathur
Department of Electrical Engineering and Computer Science
University of Toledo, Toledo, OH, USA
akshay.mathur@utoledo.edu

Ahmad Y Javaid
Department of Electrical Engineering and Computer Science
University of Toledo, Toledo, OH, USA
ahmad.javaid@utoledo.edu

***Abstract*— The advent of HTML 5 revives the life of cross-site scripting attack (XSS) in the web. Cross Document Messaging, Local Storage, Attribute Abuse, Input Validation, Inline Multimedia and SVG emerge as likely targets for serious threats. Introduction of various new tags and attributes can be potentially manipulated to exploit the data on a dynamic website. The XSS attack manages to retain a spot in all the OWASP Top 10 security risks released over the past decade and placed in the seventh spot in OWASP Top 10 of 2017. It is known that XSS attempts to execute scripts with untrusted data without proper validation between websites. XSS executes scripts in the victim's browser which can hijack user sessions, deface websites, or redirect the user to the malicious site. This paper focuses on the development of a browser extension for the popular Google Chromium browser that keeps track of various attack vectors. These vectors primarily include tags and attributes of HTML 5 that may be used maliciously. The developed plugin alerts users whenever a possibility of XSS attack is discovered when a user accesses a particular website.**

Keywords*— *cross-site scripting, tags, attributes, attack vectors, extensions.

I. Introduction

Although XSS has existed for a long time, it has been ranked consistently in OWASP Top 10 security issues cheat sheet [5]. Cross-site scripting is a security vulnerability of web pages where a hacker injects malicious JavaScript code into the website to redirect the user to a malicious site. It mainly occurs when the application includes untrusted data on its webpage. For instance, this vulnerability takes place when GET variables are printed without checking or filtering the content. The comments section in a web page is the most vulnerable part which can be used to construct an attack. This paper presents a technique to avoid XSS attack by creating an extension for a browser. An extension provides additional functionality to the browser. Extensions are available in the browser store where it can be downloaded and added to the browser. The first and challenging part of this paper is to create a repository which contains tags and attributes that serve as attack vectors. Secondly, an extension is created by linking the repository created in the first step. The functionality of this extension is to work in the background looking for suspicious activities and alert the users straight to the web browser. This extension is created using technologies like HTML, CSS, JavaScript, and JSON. For example, `<video>` and `<source>` tag can be used to construct an attack. The `<video>` tag loads and play the video. When there is no target video resource, it will fail, and the scripts attached to the event function will be executed [1].

<video><source on error="alert(1)"></video>

A vector displaying *form* and *formaction* capabilities for form hijacking outside the actual form [6].

<form id="test"></form><button form="test" formaction="javascript:alert(1)">X</button>

Allowing users to submit markup containing the form and *formaction* attributes or transform them to bogus attributes is a serious threat. The best practice is to avoid using *id* attributes for forms and submit buttons.

Table 1. Sample HTML5 Security Cheat Sheet [6]

Self executing focus event via autofocus	<input onfocus=write(1) autofocus>
Self executing blur event via autofocus competition	<input onblur=write(1) autofocus><input autofocus>
JavaScript execution via <Video> poster attribute	<video poster=javascript:alert(1)//></video>
Self executing JavaScript via <BODY> onscroll autofocus	<body onscroll=alert(1)> br>... <input autofocus>
Form surveillance with onformchange , onforminput and form attributes.	<form id=test onforminput=alert(1)><input></form><button form=test onformchange=alert(2)>X</button>
JavaScript execution via <video> and <source> tag	<video onerror="alert(1)"><source></source></video>
Xss via formaction- requiring user interaction	<form><button formaction="javascript:alert(1)">X</button>
Passive javascript execution via <BODY> and oninput attribute	<body oninput=alert(1)><input autofocus>

II. COLLECTING ATTACK VECTORS

The foremost task in building the extension is to collect the attack vectors. These vectors are manipulated to execute scripts in the victim's browser to hijack sessions. These attack vectors are collected from internet based on XSS cheat sheet and HTML 5 security cheat sheet and stored in the repository [4]. The attack vectors are categorized into three - TAGS, ATTRS and SVG. TAGS contain <video> <audio> and <embed> in HTML5. ATTRS contains autofocus and formaction. SVG contains the attack vectors supported by HTML5. JavaScript pseudo-protocol, base64 encoding, etc. are the transform techniques used for these attack vectors to bypass the filtering mechanism. In this way, a large number of attack vectors can be collected and embedded to the extension [1].

Table 2. Sample Attack Vectors

TAGS	ATTRS	SVG
<EMBED SRC=“” > type=“” AllowScriptAccess =”always” </EMBED>	<body oninput=alert(1) ><input autofocus>	<svg xmlns=“”><g onload+”JavaScript:alert(1)” ></g></svg>
<video onerror=”alert(1)”> <source></source> </video>	<input onfocus=alert(1) autofocus>	<svg onload=” JavaScript:alert(1)” xmlns=“”></svg>
<video src=“” onerror=alert(1)	<input onblur=alert(1) autofocus><input autofocus>	<svg xmlns=“”><a xmlns:xlink=“” xlink:href=”JavaScript:alert(1)” ><rect width=“”height=“” fill=” white”/></a></svg>
<video><source onerror=”alert(1)”	<form><button formaction=“”> X</button>	

III. RELATED WORK

Two research findings serve the basis of this paper. Dr.Zhang in his research paper titled "Cross-site scripting attack in the social network –APIs" provided a systematic study on analysis XSS flaws within social API and detected various security issues such as tainted API responses, inconsistent handling of user input data and incorrect API responses. He designed a tool to analyze the design and implementation flaws in RESTful APIs in social networks [3]. Another finding by Dr. Dong that detects XSS attack on the webmail systems by creating a tool using java. The tool is implemented by importing Java Mail Development Kit. They collect attack vectors and store in the repository. The key to this approach is to inject attack vectors into the checkpoints. He identified email title, email body, attachment title, attachment name and sender name as checkpoints. The tool generates test emails and sends to target mailboxes. But these findings are not a comprehensive solution to XSS attack on the web since it focuses only on APIs used in social networks and webmail systems. The approach proposed in this paper targets API and webmail systems by not creating an external tool but by adding an extension to the browser.

Table 3. DOM Based Attack Vectors

KEYWORDS	EXAMPLE
document	document.getElementById, document.getElementByName
location	location.herf, location.replace
window	window.location, window.history
history	history.back, history.forward, history.go

IV. ATTACK PATTERNS

The next important step is to identify the attack patterns.The attack vectors are transformed by three techniques. Firstly, changing special characters into newline, tabs, etc. Secondly, changing the double quotes into the single. Thirdly, attack vectors are encoded by decimal coding. The extension works by identifying attack patterns and alert the users at the browser. This paper describes three different attack patterns. They are Direct attack patterns, Multi-format attack patterns, and DOM detection patterns.

A. Direct Attack Patterns: In Direct attack patterns**,** attack vectors are obviously determined. For example,

<body onscroll=alert(1)>
<br><br><br><br><br><br><br><br>
<input autofocous>

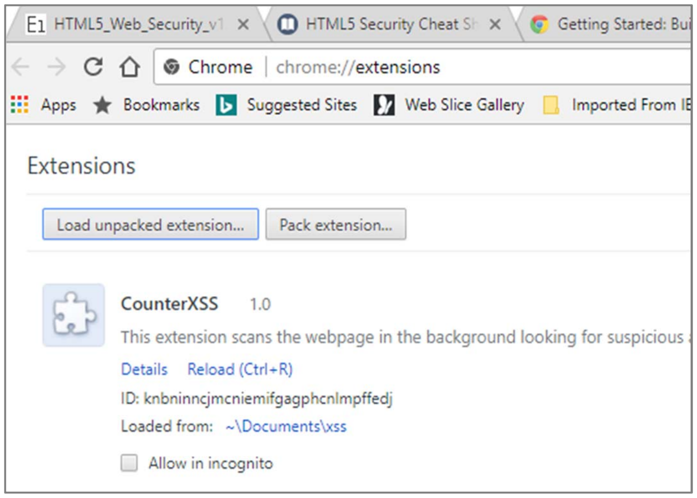
B. Multiple Format Attack patterns: Multiple format attack patterns include direct attack patterns as well as attributes that can be more comprehensively exploited. For example, *src* attribute indicates a source URL, commonly used in many tags.

<img src="http://whatattacker.com/cookies.php?id="+localStorage .getItem('SessionID')+"'>");
</img>

C. DOM Attack Patterns: In DOM detection pattern, Document Object Model, like *onerror, onload* etc., is the extensive use of automatic execution of the script. It is usually found in the input field that can implement XSS attacks without victim's approval [2].

V. BROWSER EXTENSION DEVELOPMENT

A browser extension is a small program that adds extra features and functionalities to the browser. It is developed using HTML, CSS, JavaScript, and JSON. Different file requirements for creating an extension are discussed here. The first and important file in creating an extension is to create a *Manifest.json* file that is basic metadata of the extension. The *Manifest.json* file contains details like name and version. It also indicates specific aspects of extension's functionality. An HTML page is created that pops up when you click the extension icon and a JavaScript page to control HTML and CSS. JavaScript code is placed in a separate file. The extension proposed in this paper is developed to identify these attack patterns and alert the users. This extension works in the background looking for all the three support patterns and alerts the user when one or more patterns are found. The extension is built for the Google Chromium browser, as shown in Figure 1. Once the required files are created, it can be added to the browser by just clicking “Load unpacked extensions” button in the extensions option, and enabling developer mode, in the browser and attach the required files. After attaching the files, the extension appears in the right corner of the toolbar of the browser. The extension starts working automatically when the browser is loaded and also works whenever a new tab opens in the browser, as shown in Figure 2. A particular web page can also be scanned individually by clicking scan manually option in the extension page. Corresponding results are obtained and displayed in the same plugin pop-up window, as shown in Figure 3.


Extensions

Load unpacked extension... Pack extension...

CounterXSS 1.0

This extension scans the webpage in the background looking for suspicious

Details Reload (Ctrl+R)

ID: knbninncjmcniemifgagphcnlmpffedj

Loaded from: ~\Documents\xss

Allow in incognito

Figure 1. Loading the unpacked extension in the chrome browser

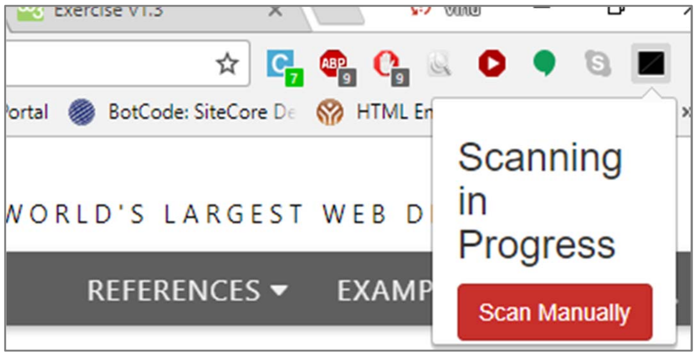
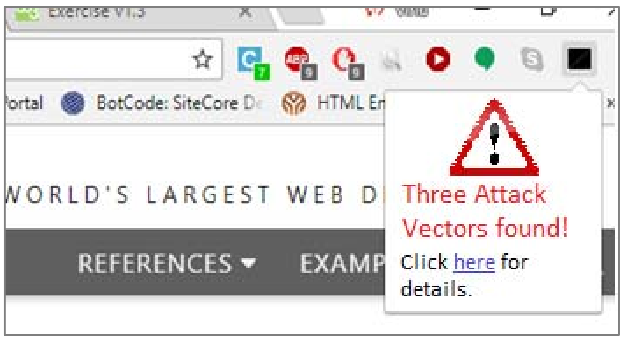


Figure 2. Extension in use

VI. Conclusion

In this paper, we presented an approach for creating a browser extension to counter cross-site scripting attack**.** HTML 5 tags and attributes have brought several new security challenges to the web. The key to this paper is collecting attack vectors and


Three Attack Vectors found!

Click here for details.

Figure 3. Example results obtained from the extension

storing them in the repository**.** These attack vectors are collected from XSS cheat sheet and HTML 5 cheat sheet**.** This paper also presents an analysis of different attack patterns and find them using the browser extension. This extension is built using HTML, CSS, JavaScript, and JSON. It works in the background and alerts the user right at the browser if it finds one or more attack patterns. The extension starts working automatically whenever a new tab is opened in the browser A particular web page can also be scanned individually by clicking scan manually option in the extension page.

References

[1] G. Dong, Y. Zhang, X. Wang, P. Wang, and L. Liu, “Detecting cross-site scripting vulnerabilities introduced by HTML5,” International Joint Conference on Computer Science and Software Engineering (JCSSE), 2014.

[2] C.-H. Wang and Y.-S. Zhou, “A new Cross-site Scripting Detection Mechanism Integrated with Html 5 and Cors properties by using Browser Extensions,” International Computer Symposium, 2016.

[3] Y. Zhang, Q. Liu, Q. Luo, and X. Wang, “XAS: Cross-API scripting attacks in social ecosystems,” Science China Information Sciences, vol. 58, no. 1, pp. 1–14, 2014.

[4] XSS (Cross Site Scripting) Cheat Sheet, 2011.http://hackers.org/xss.html

[5] Category:OWASP Top Ten Project – OWASP. [Online], Available: https://www.owasp.org/index.php.php/Top10=OWASP_Top_10_for_2013. [Accessed: 01-Nov-2017].

[6] “HTML5 Security Cheatsheet What your browser does when you look away…,” HTML% Security CheatSheet. [Online]. Available: http://html5sec.org/.[Accessed: 17-Jan-2018].