

# Characterizing Insecure JavaScript Practices on the Web

Chuan Yue  
 Department of Computer Science  
 The College of William and Mary  
 Williamsburg, VA 23187, USA  
 cyue@cs.wm.edu

Haining Wang  
 Department of Computer Science  
 The College of William and Mary  
 Williamsburg, VA 23187, USA  
 hnw@cs.wm.edu

## ABSTRACT

JavaScript is an interpreted programming language most often used for enhancing webpage interactivity and functionality. It has powerful capabilities to interact with webpage documents and browser windows, however, it has also opened the door for many browser-based security attacks. Insecure engineering practices of using JavaScript may not directly lead to security breaches, but they can create new attack vectors and greatly increase the risks of browser-based attacks. In this paper, we present the first measurement study on insecure practices of using JavaScript on the Web. Our focus is on the insecure practices of JavaScript inclusion and dynamic generation, and we examine their severity and nature on 6,805 unique websites. Our measurement results reveal that insecure JavaScript practices are common at various websites: (1) at least 66.4% of the measured websites manifest the insecure practices of including JavaScript files from external domains into the top-level documents of their webpages; (2) over 44.4% of the measured websites use the dangerous `eval()` function to dynamically generate and execute JavaScript code on their webpages; and (3) in JavaScript dynamic generation, using the `document.write()` method and the `innerHTML` property is much more popular than using the relatively secure technique of creating script elements via DOM methods. Our analysis indicates that safe alternatives to these insecure practices exist in common cases and ought to be adopted by website developers and administrators for reducing potential security risks.

## Categories and Subject Descriptors

H.3.5 [Information Storage and Retrieval]: Online Information Services—*Web-based services*; I.7.2 [Document and Text Processing]: Document Preparation—*Scripting languages*; K.6.5 [Management of Computing and Information Systems]: Security and Protection—*Unauthorized access*

## General Terms

Experimentation, Languages, Measurement, Security

## Keywords

JavaScript, execution-based measurement, security, same origin policy, AST tree matching, Web engineering

Copyright is held by the International World Wide Web Conference Committee (IW3C2). Distribution of these papers is limited to classroom use, and personal use by others.

WWW 2009, April 20–24, 2009, Madrid, Spain.  
 ACM 978-1-60558-487-4/09/04.

## 1. INTRODUCTION

Security is an important aspect of Web engineering, and it should be taken into serious consideration in the development of high quality Web-based systems [5, 15, 19, 21, 23]. In many cases, however, security does not receive sufficient attention due to the complexity of Web-based systems, the ad hoc processes of system development, and even the fact that many designers or developers lack security knowledge on Web development techniques. It is not a surprise therefore, that website security breaches are common [7] and Web applications are more susceptible to malicious attacks than traditional computer applications [27].

Browser-based attacks have posed serious threats to the Web in recent years. Exploiting the vulnerabilities in Web browsers [6, 25] or Web applications [11, 14], attackers may directly harm a Web browser's host machine and user through various attacks such as drive-by download [20, 24, 28], cross-site scripting [10, 37], cross-site request forgery [2, 46], and Web privacy attacks [4, 12]. Attackers may even use browsers to indirectly launch large-scale distributed attacks against Web servers [17] or propagate Internet worms [18].

Most of these browser-based attacks are closely tied with JavaScript, which is an interpreted programming language most often used for client-side scripting. JavaScript code embedded or included in HTML pages runs locally in a user's Web browser, and it is mainly used by websites to enhance the interactivity and functionality of their webpages. However, because JavaScript is equipped with a powerful and diverse set of capabilities in Web browsers [9], it has also become the weapon of choice for attackers.

Modern Web browsers impose two restrictions to enforce JavaScript security: the *sandbox* mechanism and the *same-origin* policy. The former limits JavaScript to execute only in a certain environment without risking damage to the rest of the system, while the latter prevents JavaScript in a document of one origin from interacting with another document of a different origin [9, 45]. Unfortunately, most JavaScript-related security vulnerabilities are still the breaches of either of these two restrictions [40]. Some of these vulnerabilities are due to Web browser flaws, but the majority of them have been attributed to the flaws and insecure practices of websites [46, 48].

A great deal of attention has been paid to the JavaScript-related security vulnerabilities such as cross-site scripting [10, 29, 38, 46, 48] that could directly lead to security breaches. However, little attention has been given to websites' insecure practices of using JavaScript on their webpages. Similar to websites' other insecure practices such as using the

















