Code-injection Vulnerabilities in Web Applications – Exemplified at Cross-site Scripting

Code-injection Verwundbarkeit in Web Anwendungen am Beispiel von Cross-site Scripting

Martin Johns, SAP Research, Karlsruhe

Summary Cross-site Scripting (XSS) is one of the most prevalent vulnerability types that affect Web applications. This article provides an overview of a dissertation, which addresses the problem XSS as a whole: It starts with a systematic deduction of causes and consequences of XSS, proceeds with presenting countermeasures to mitigate potential XSS-based attacks, and finally provides a type-based methodology that guarantees the creation of XSS-free applications. ►►► **Zusammenfassung** Cross-site Scripting (XSS) ist eine der häufigsten Verwundbarkeitstypen im Bereich der Web Anwendungen. Die in diesem Artikel vorgestellte Dissertation behandelt das Problem XSS ganzheitlich: Basierend auf einer systematischen Erarbeitung der Ursachen und potentiellen Konsequenzen von XSS, wird zunächst eine Methodik vorgestellt, die das Design von dynamischen Gegenmaßnahmen zur Angriffseingrenzung erlaubt. Weiterhin, um das unterliegende Problem grundsätzlich anzugehen, wird ein Typ-basierter Ansatz zur sicheren Programmierung von Web Anwendungen beschrieben, der zuverlässigen Schutz vor XSS Lücken garantiert.

Keywords D.3.4 [Software: Programming Languages: Processors]; D.2 [Software: Software Engineering]; Security, Code Injection, Web Application ►►► **Schlagwörter** Web Anwendung, Sicherheit, Angriffsklassifikation, Angriffsabwehr, sichere Programmierung

1 Introduction

The Web has won. No other plattform for distributed applications can rival the Web's ubiquity and flexibility: Web applications cover all imaginable application types, e. g., e-commerce shops, ERP systems, online banking, social networks, office applications, image manipulation applications, games, or front-ends for hardware appliances, such as DSL modems or internet routers. In the same pace as the Web application paradigm's importance has risen, the prevalence and severity of Web application vulnerabilities have increased. One of the most common vulnerabilities that affect Web applications is *Cross-site Scripting (XSS)* [1].

XSS is an attack which enables the adversary to execute arbitrary JavaScript within the victim's Web browser when the vulnerable Web application is accessed. This JavaScript is executed in the victim's current execution context, hence allowing the attacker to conduct actions that abuse the victim's current authentication state [2].

2 Understanding XSS and XSS Payloads

A thorough understanding of the underlying mechanisms of XSS attacks is indispensable to assess all potential defensive approaches. Therefore, the first part of the thesis explores the technical aspects of the web application

[0] The dissertation is entitled "Code Injection Vulnerabilities in Web Applications – Exemplified at Cross-site Scripting". The entire text is available at http://www.opus-bayern.de/uni-passau/volltexte/2011/2362/. The examiners were Prof. Dr. Joachim Posegga (University of Passau) and Prof. Dr. Dieter Gollmann (TU Hamburg-Harburg). The dissertation has been recommended to the GI-Dissertation Award 2009 by the University of Passau.

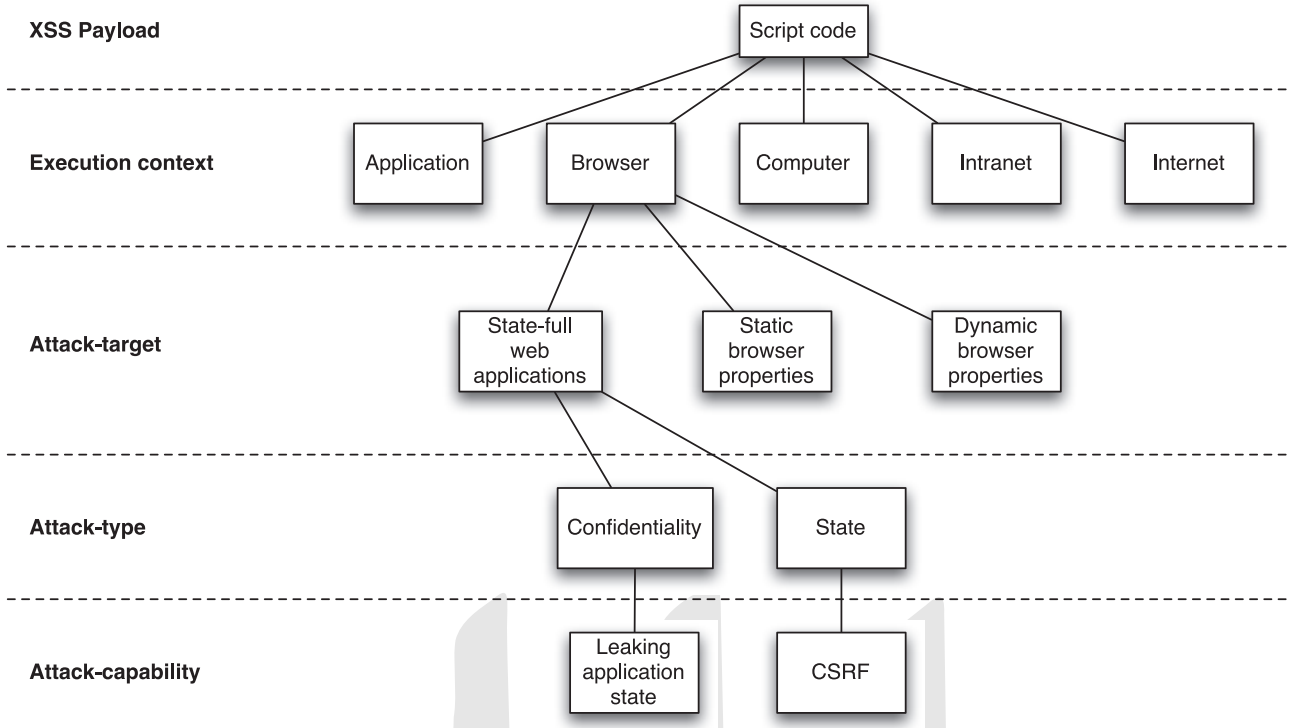

Figure 1 Classification of XSS Payloads (exemplified).

paradigm, the causes of XSS vulnerabilities, the specific methods of exploiting such issues, and the malicious capabilities which an adversary may gain by the exploitation.

For this purpose, we present a comprehensive survey of documented XSS attacks [9]. Based on this survey, we deduct a set of general attack techniques, which are the basic building blocks of any XSS attack, and introduce a comprehensive and systematic classification of potential XSS Payloads. Our proposed classification is based on dividing the potential actions of a given JavaScript according to a disjunct set of execution-contexts. This enabled us to group individually reported attacks into larger classes and to identify the set of existing payload targets, such as the affected web application, the victim's computer, or intranet resources (see Fig. 1).

Based on the results of this thorough examination of XSS, we can deduce two general directions towards solving the discussed issues:

- Designing dedicated countermeasures to disarm specific payload classes.
- Introducing methods towards removing the underlying XSS issues by changing the process of developing web applications.

The following two parts of the thesis present our approaches in respect to these two general areas.

3 Mitigating XSS Exploitation

As soon as the attacker is able to execute his JavaScript in the victim's browser, his activities are unrestricted in respect to the attack payloads which have been identified in the survey. Consequently, steps have to be taken to mitigate the identified attack classes, even in the existence of XSS flaws.

As long as the process of web application development has not reached a state in which XSS problems are only rarely encountered, this general approach is valid to establish a second line of defense.

The techniques proposed in the thesis all share the same underlying methodology: They aim to disarm XSS Payloads by selectively depriving the adversary of capabilities, which are required to successfully execute the targeted attack.

This is achieved by transparently modifying the execution environment of the web application without requiring changes in the actual application.

Within the thesis, countermeasure for the three prevailing XSS Payload types have been designed: *Session Hijacking*, *Cross-site Request Forgery*, and *Intranet Attacks*:

- **Session Hijacking** [5]: First, we closely examine the distinct methods of XSS-based Session Hijacking which have been identified in the payload survey. Based on this analysis, we propose three technical measures, each tailored to disarm one of these possible session hijacking attacks. A combination of our three methods prevents all session hijacking attempts despite existing XSS problems.
- **Cross-site Request Forgery (CSRF)** [6]: To mitigate CSRF Payloads, we utilize a similar methodology as in the case of Session Hijacking: First, we closely analyse the underlying mechanisms that enable CSRF attacks. Then, we introduce changes in the vul-

nerable authentication tracking mechanisms which devoid the adversary from successfully launching CSRF attacks.

- **Intranet Attacks [7]:** Finally, we address the class of attacks that target intranet resources. Due to an initial examination of the attack class, we deduct three potential countermeasures (in addition to the practice of disabling JavaScript completely). We discuss the advantages and drawbacks of each method and conduct a comparison of the four methods. Based on this discussion, the most promising approach is implemented and practically evaluated.

For details on the individual countermeasures, please refer to the thesis or the associated publications.

4 Enforcing Secure Code Generation

While mitigation of attacks, as described in the previous section, is an important pillar of any defense-in-depth strategy, it is of high importance to address the underlying root cause of XSS: The majority of all XSS issues are caused by insecure programming. Thus, a careful examination of the underlying coding practices is necessary to establish possible fundamental solutions.

On closer examination, it becomes apparent, that XSS flaws are actually only a subtype of a larger vulnerability-class: The class of *string-based code injection vulnerabilities*. Other members of this class are for instance *SQL Injection*, *Directory Traversal*, or *Remote Command Injection*. For this reason, we analyse the root causes of the more general case:

String-based code injection occurs in situations where a program dynamically assembles computer language code for further usage. This code assembly is done using the string datatype. Code which is created this way, is subsequently passed to other parsers during run-time to be immediately interpreted. String-based code injection occurs because programmers insecurely mix code-syntax with data-values during this process. In such situations, the adversary is capable to trick the program into including data-values which contain syntactic elements into the code assembly, hence, altering the semantics of the resulting computer code.

To solve this problem, we propose a strong separation between data and code during dynamic syntax assembly. For this purpose, we propose definitions of the concepts *data* and *code* that are applicable to string-based code assembly. Then, we analyse the structure of selected computer languages. This enables us to classify individual language elements to represent either *data*- or *code*-elements (see Fig. 2).

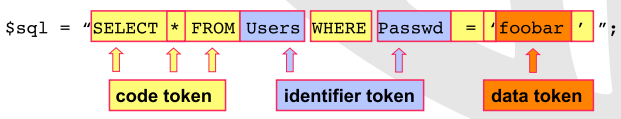

Figure 2 Data/Code separation for string-based code assembly.

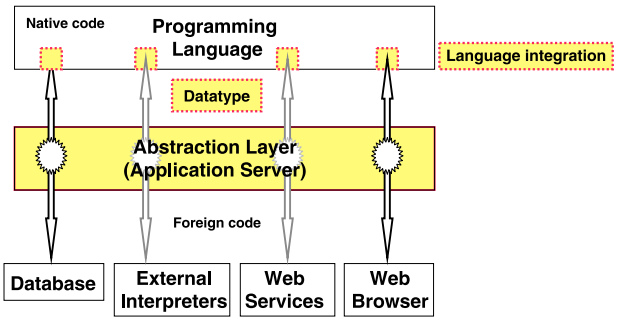

Figure 3 Abstraction layer.

Based on these results, we successively develop a novel, language-based method for dynamic code assembly. The central concept of our approach is to exchange the common, inherently insecure code assembly practices with a secure methodology. More precisely, we introduce a novel datatype, the Foreign Language Encapsulation Type (FLET) which replaces the string type for code assembly. The FLET enforces a strict separation between *data*- and *code*-elements, hence, rendering programming mistakes which lead to data/code-confusion impossible. To substantiate this security claim, we provide a formal type theoretical proof which is based on the Biba integrity model [3] and Volpano/Smith's information flow formalization [4].

Furthermore, to ensure mandatory usage of the FLET semantics, we propose the removal of all direct interfaces to external interpreters. Instead, we introduce an abstraction layer mechanism which provides a FLET-based interface for secure code-communication (see Fig. 3). To verify the usability of our approach, we show how to practically implement our concepts for the J2EE application server [8].

5 Conclusion

The thesis addresses the problem XSS as a whole: For one, the root causes and potential malicious consequences of this vulnerability type are deducted and comprehensively presented. Furthermore, the thesis shows how to handle the problem from a defensive point of view, covering both reactive countermeasures as well as preventive methods to ensure XSS-free applications through security by construction.

However, the web application paradigm is still evolving. Both JavaScript and HTML are under active development. Web browsers recently started to implement HTML5, the next major version of the language. New language elements and extended capabilities, such as cross-domain HTTP requests or persistent client-side storage, may grant the adversary new capabilities. Therefore, existing and proposed countermeasures have to be continuously reevaluated whether they still function given the current state of the technology. Also, the novel capacities may lead to the development of currently unknown XSS Payloads.

Nonetheless, the methodologies discussed in the thesis remain valid for new attacks: For one, the underlying approach of our payload classification (segmentation of execution-contexts and identification of attack targets through URL schema iteration) is independent from actual language features and, hence, can be applied to assess freshly discovered payload types. Furthermore, our general methodology to develop payload-specific mitigation can be utilized to create suiting countermeasures.

Also, the attack surface of XSS attacks is directly related to the number of existing XSS vulnerabilities in deployed applications. Thus, a wide adaption of our FLET-based technique for reliably secure foreign code assembly would cause a significant reduction of this attack surface.

Consequently, the thesis' contributions can provide crucial leverage to address the pressing problem of XSS.

References

[1] St. Christey and R. A. Martin. Vulnerability Type Distributions in CVE, Version 1.1, May 2007. Online http://cwe.mitre.org/documents/vuln-trends/index.html.

[2] J. Grossman, R. Hansen, P. Petkov, and A. Rager. *Cross Site Scripting Attacks: XSS Exploits and Defense*. Syngress, 2007.

[3] K. J. Biba. Integrity Considerations for Secure Computer Systems. Report MTR-3153, Mitre Corporation, April 1977.

[4] D. M. Volpano, G. Smith, and C. Irvine. A sound type system for secure flow analysis. In: *Journal of Computer Security*, 4:167–187, 1996.

[5] M. Johns. SessionSafe: Implementing XSS Immune Session Handling. In: *Proc. of the European Symp. on Research in Computer Security (ESORICS 2006)*, LNCS 4189, pages 444–460, Springer, Sep 2006.

[6] M. Johns and J. Winter. RequestRodeo: Client Side Protection against Session Riding. In: *Proc. of the Open WEb Application Security Project (OWASP) Europe Conf.*, May 2006. Online: https://www.owasp.org/images/4/42/RequestRodeo-MartinJohns.pdf.

[7] M. Johns and J. Winter. Protecting the Intranet Against "JavaScript Malware" and Related Attacks. In: *Proc. of the 4th Int'l Conf. on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2007)*, LNCS 4579, pages 40–59. Springer, July 2007.

[8] M. Johns, Chr. Beyerlein, R. Giesecke, and J. Posegga. Secure Code Generation for Web Applications. In: *Proc. of the 2nd Int'l Symp. on Engineering Secure Software and Systems (ESSoS '10)*, LNCS 5965, pages 96–113. Springer, Feb 2010.

[9] M. Johns. On JavaScript Malware and Related Threats – Web Page Based Attacks Revisited. In: *Journal in Computer Virology, Springer Paris*, 4(3):161–178, Aug 2008.

Received: June 20, 2011

Dr. Martin Johns is working as Senior Researcher in the security and trust group within SAP Research, where he is currently leading the Web application security team. Before joining SAP, Martin studied Mathematics and Computer Science at the Universities of Goettingen, Santa Cruz (CA) and Hamburg where he received his diploma in 2003. During the 1990ties and the early years of the new millennium he earned his living as a software engineer in German companies (including Infoseek Germany and TC Trustcenter). 2005 he joined the "Security in Distributed Systems" group at the University of Hamburg to conduct software security research. Subsequently, he worked as research assistant at the University of Passau where he finished his Ph.D thesis on Web security in 2009.

Address: SAP Research, Vincenz-Priessnitz-Straße 1, 76131 Karlsruhe, Germany, Tel.: +49-6227-752547, Fax: +49-6227-7844618,
e-mail: martin.johns@sap.com

Weltweit anerkanntes Standardwerk

David Patterson/ John LeRoy Hennessy

Rechnerorganisation und Rechnerentwurf

Die Hardware/Software-Schnittstelle

2011 | XXIII | 724 S. | Br.
ca. € 59, 80
ISBN 978-3-486-59190-3

Mit der deutschen Übersetzung zur dritten Auflage des amerikanischen Klassikers »Computer Organization and Design« ist das Standardwerk zur Rechnerorganisation wieder auf dem neusten Stand – David A. Patterson und John L. Hennessy gewähren die gewohnten Einblicke in das Zusammenwirken von Hard- und Software, Leistungseinschätzungen und zahlreicher Rechnerkonzepte in einer Tiefe, die zusammen mit klarer Didaktik und einer eher lockeren Sprache den Erfolg dieses weltweit anerkannten Standardwerks begründen.

»Hochaktuell, inspirierend geschrieben, reichhaltig ausgestattet. Das Standardwerk zur Rechnerorganisation, das keine Wünsche offen lässt.«
(Prof. Dr.-Ing. Märtin, FH Augsburg)

Umfangreiches Zusatzmaterial (zusätzliche Aufgaben samt Lösungen, Werkzeuge mit Tutorien etc.) steht auf der beiliegenden CD zur Verfügung. Für Hardwarespezialisten und Softwareentwickler, für Theoretiker ebenso wie für Praktiker.

Dr. David Patterson ist Professor für Computer Science an der University of California, Berkeley.

John LeRoy Hennessy ist Präsident der Stanford University und Professor für Elektrotechnik und Informatik.