Security Analysis Regarding Cross-Site Scripting on Internet Explorer

[Extended Abstract]

Adriana Neagoş
Babeş Bolyai University
Kogalniceanu str. 1
Cluj-Napoca, Romania
naie1000@scs.ubbcluj.ro

Simona Motogna
Babeş Bolyai University
Kogalniceanu str. 1
Cluj-Napoca, Romania
motogna@cs.ubbcluj.ro

ABSTRACT

The purpose of this paper is to provide an exact evaluation of cross site scripting vulnerabilities on security, as an important factor of software quality. Since, this kind of risks are dependent on the browser, the study takes into consideration three versions of Internet Explorer, and uses an established scoring system, the Common Vulnerability Scoring System, to measure their impact.

Categories and Subject Descriptors

H.4.m [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*Complexity measures, Performance measures*

General Terms

Security

Keywords

Cross site scripting, security

1. INTRODUCTION

Software development focuses on delivering applications with minimal resources. Architects and developers want to produce and deploy applications, ready to be executed, in a short amount of time and only with the strictly necessary resources (people, software components and hardware). Software quality is not always set as an important issue, and tend to be neglected, especially when working against time or with a limited team. Several studies (NASA, IBM) have lead to an important conclusion: *Improving quality reduces development costs.*

Over the recent years web applications tend to replace desktop applications, since such an approach makes them more accessible. In these conditions, software quality factors are changing their importance, and *security* becomes an important factor not only in the evaluation of an application, but, more importantly, in assuring a reliable operation of the software. In consequence, a lot of research, both in academia and industry, focuses on studying risks, vulnerabilities, and attacks to software security. Open Web Application Security Project (OWASP) [7] is such an initiative, and maintains and updates a list of top 10 Security Risks. Cross-Site Scripting (XSS) is the second as importance, and considered as having an average exploitability and a high degree of occurrence. There are a lot of cases in which automatic tools detect this risk in an easy way, but, however, there are some situations, generated by new technologies and browser characteristics, that make detection more difficult.

The purpose of this article is to present an in-depth analysis of detecting cross-site scripting vulnerabilities and their impact on software security factor. The rest of the paper is organized as follows: section 2 presents some of the existing related work. Section 3 contains an analysis of XSS vulnerabilities and of the security policies proposed by different browsers, while the next section is dedicated to our evaluation of XSS vulnerabilities for Internet Explorer, and in the end we draw some conclusions and future directions.

BCI'12, September 16–20, 2012, Novi Sad, Serbia.

Local Proceedings also appeared in ISBN 978-86-7031-200-5, Faculty of Sciences, University of Novi Sad.

2. RELATED WORK

A lot of research has been carried out in the field of XSS vulnerabilities. Most of them focuses on studying pattern attacks, evaluating risks and proposing solutions to prevent them [10], [11], [12].

Security Evaluation and Measurement: Besides the approaches carried out at major software companies, such Microsoft, IBM, Apple a.s.o., there are two important contributions to assesing security vulnerabilities and proposing metrics to evaluate their impact on software quality:

• Computer Emergency Center (CERT) at Carnegie Melon University[1] with results in risk analysis, based on a tactical andon a systematic approach, and security measurement, that are integrated in IMAF (Integrated Measurement and Analysis Framework) [8].

• Common Vulnerability Scoring System (CVSS) [9] that developed a framework that supports scoring of security vulnerabilities.

3. ANALYSIS OF CROSS SITE SCRIPTING VULNERABILITIES

XSS is performed by injection of code (Javascript, ActiveX, Silverlight, Flash) that is executed by a browser. This kind of code should be executed under sandboxing mechanism which means that only a set of operations should be

[1]www.cert.org

performed, but it is not enough. Michael Howard said that "All input is evil until proven otherwise. That!s rule number one" [13], but in case of XSS not only inputs, but also outputs must be validated.

There are 3 types of XSS:

- non-persistent or reflected: is performed when there is no proper validation of user input through GET or POST requests and the response page is returned immediately and is spread generally by email via malicious urls.
- persistent: happens when the infected code is stored in the database and it is a regular threat to chat software or application including different posts. User does not access malicious links, just regular browsing can result into being robbed of information.
- DOM-based: results from dynamically-computed data, which means that the browser is manipulated to render DOM elements controlled by an attacker.

Programming languages provide in-built functions that perform this kind of filters, but even Microsoft states regarding their ASP.NET method ValidateRequest that one should not rely only on this type of validation because unfortunately it is not 100 percent secure. Recent attacks prove this. Not only ASP.NET functions have security lacks. Parse_url is a function in PHP that verifies malformed urls. The function works correctly in most cases except if whitespaces are inserted. This was the vulnerability exploited on April 2011, on Facebook, when a malicious video was posted [14] or on CNN when urls inserted in ad networks were source of this attack. Other exploits were done also on The New York Times, on Twiter, e-Bay or on Fox News [15].

3.1 Security Policies

XSS is the one security field that does not depend on the type of connection: encrypted or unencrypted, but is closely related to portability and mainly browser compatibility. Because it is rendered by different browsers, the display of a web page can by slightly different, and so its gate of access. *Same origin policy* is called the policy adopted against browser-side languages that does not allow "access to most methods and properties across pages on different sites". It is implemented by nearly each browser, but it does not guarantee complete security. In addition, modern browsers implemented several security policies that block an attacker to gain access on a client machine. Firefox and Opera are known as relatively secure, but in this paper I would like to refer to Internet Explorer, which is considered very vulnerable. A simple example is for instance when uploading text files through IE: if HTML content is inserted in the file, it doesn't treat it as plain text, but it interprets it as HTML.

Internet Explorer 6 introduced HttpOnly cookie attribute which intended to protect against retrieving information through document.cookie and Internet Explorer 7 made sure this information was not available in the response header through XMLHttpObject. "HttpOnly cookies don't make you immune from XSS cookie theft, but they raise the bar considerably" [2]

Starting with Internet Explorer 8, there is introduced a new very controversial browser filter for XSS. Still, Michael Brooks describes it as vulnerable and users claim that it also considers safe pages as potentially dangerous [4]. Google disables it by setting the X-XSS-Protection in the header to 0 or it can be turned off from the browser security tab.

In March 2011, together with the release of version 4, Firefox proposed the adoption of a new layer to enforce XSS protection called the *Content Security Policy*. This framework is still not implemented yet on other browsers, but Microsoft claims that it will be a feature of Internet Explorer 10.

Runtime protection methods should also be taken into consideration, even if they affect the performance of the application. *Web application firewalls*(WAF's) monitor the communication flow across the network and therefore they inspect messages for Javascript and can enforce a set of rules in order identify and block XSS attacks that would not reach no more the backend. Example of such applications are Cisco ACE Web Application Firewall [5], NetScaler App Firewall [6] or Barracuda Web Application Firewall [3]. Most WAF's implement the *Intercepting Filter* pattern or include one or more implementations in their overall architecture. One can also add filters to an application at deployment when implemented as a Web server plug-in or when activated dynamically within an application configuration file.

Moreover, Microsoft offers an *Anti-Cross Site Scripting Library* [1] and OWASP advises programmers to use an API: *ESAPI* (The OWASP Enterprise Security API) [7] which is an open source web application security control library.

Users should be also educated to avoid XSS exploits. Avoiding awkward links, paying attention to redirections or for instance turning off the HTTP TRACE can prevent the stealing of cookies.

Regardless the variety of prevention methods new exploits continue to attack web applications and it's our duty to keep on protection against the known or unknown security flaws.

4. CVSS SCORES FOR XSS VULNERABILITIES

Our case study consists in computing the CVSS vulnerability scores for some XSS related vulnerabilities reported by Microsoft. CVSS or the Common Vulnerability Scoring System is an open framework that provides a numerical score by taking into consideration base, temporal or environmental properties of a certain vulnerability. The computatiopn is performed according to the formula given in [9] and using the calculator available at http://nvd.nist.gov/cvss.cfm?calculator&adv&version=2. Each of the three metric groups has its own characteristics and contains a set of metrics. Base metric group describes the fundamental characteristics of a vulnerability and is composed of the related exploit range, the attack complexity, the needed authentication level and the integrity, availability and confidentiality impact. Temporal are the metrics influenced by time passing, meaning the availability of exploit, the remediation level and the report of confidence. Environment has also an impact when computing the score; environmental factors are the collateral damage potential, the target distribution and the confidentiality, integrity and availability requirement.

When talking about XSS exploits some of these metrics remain constant because of the type of this vulnerability. The *related exploit range* or the access vector is the *network*, because the attack is widely spread over the internet and the *access complexity* is *medium*. The majority of reports

written by Microsoft describe vulnerabilities on Internet Explorer having the standard, default configurations, but it is medium and not low, because the attacker is required to have some social engineering skills in order manipulate and fool custom users to access a certain page or click a specific button or link. In order to be considered successful, the attack has to gain information or control over the client machine and for this at least *one instance of authentication* is needed. The *availability impact* metric refers to the access of the attacker over the resources, meaning bandwidth, processor, disk space and his possibility to get a total shutdown of the affected resource. In case of an XSS exploit, we consider this availability impact to be *none*; it is impossible from what we know until now for someone to access the resources using this type of vulnerability. We have intentionally skipped the confidentiality and integrity impact because they vary depending on the attack and we are focused now on the constant metrics of XSS exploits.

Moving to the temporal group, we will argue the chosen values to the metrics based on the vulnerabilities reported by Microsoft on their periodical security bulletins. We let the *exploitability* factor set to *not defined*, because officially they say that the exploitation code was not made public: "Microsoft received information about this vulnerability through coordinated vulnerability disclosure.", "Microsoft had not received any information to indicate that this vulnerability had been publicly used to attack customers when this security bulletin was originally issued". Moreover, all the vulnerabilities are *confirmed* and are reported only after an *official fix* is available.

The *damage potential* of an XSS vulnerability is according to OWASP *moderate*. We are not talking about a physical damage, but there can be significant loss of information.

Last, but not least there are the *impact requirement modifiers*. As any browser, Internet Explorer is meant to be secure. Confidentiality, integrity and availability are the three features users require for safe browsing and financial transactions.

The metrics that change depending on the XSS exploit are the confidentiality and integrity impact and the percentage of vulnerable systems. In order to see how these metrics differ we will consider three vulnerabilities URL Validation Vulnerability[2], HTML Layout Remote Code Execution Vulnerability and XSS Filter Information Disclosure Vulnerability. URL Validation Vulnerability is a critical vulnerability reported in February 2010 that appeared from incorrectly validated input. It provided the attacker access to the client machine with the same rights as the logged in user and if the attacker could reach administrative rights, he could install programs, read or change data. In this case, because remote code could be executed, the confidentiality and integrity impact is complete. Regarding the target distribution, which was Internet Explorer, it was reported as a vulnerability on IE 7 and 8 which at that time covered 72.8% of the market, so medium spread. The score in this case reaches 6.3. On August 2011, another vulnerability was reported, XSS Filter Information Disclosure Vulnerability[3]. It was performed by running malicious Javascript code in specially constructed web pages. It was reported as important, because it provided information disclosure and it was

	Confidentiality Impact	Integrity Impact	CVSS Score
Internet Explorer 7	- complete HTML Layout Remote Code Execution URL Validation Vulnerability	- complete HTML Layout Remote Code Execution URL Validation Vulnerability	 7.7 6.3
Internet Explorer 8	- complete HTML Layout Remote Code Execution URL Validation Vulnerability - partial XSS Filter Information Disclosure	- complete HTML Layout Remote Code Execution URL Validation Vulnerability - partial XSS Filter Information Disclosure	 7.7 6.3 5.5
Internet Explorer 9	- complete HTML Layout Remote Code Execution	- complete HTML Layout Remote Code Execution	7.7

Figure 1: CVSS scores for XSS vulnerabilities in Internet Explorer.

applicable only on IE 8, meaning 52% of the users having Internet Explorer. We reach a lower score 5.5 as the attacker could gain only access to information and not remote control and the confidentiality and integrity impact is partial. HTML Layout Remote Code Execution Vulnerability[4] is a more complex vulnerability, affecting IE 7, 8, 9 (94% of the market). It is related to the way Internet Explorer handles objects in memory and has a complete impact on integrity and confidentiality. In this special case, the access complexity is increased and the score reaches 7.7. Target distribution on Internet Explorer is calculated related to the date of the report taking into consideration data provided by http://www.w3schools.com/browsers/browsers_explorer.asp

The table from Figure 1 displays the resulting scores.

The first important remark is that vulnerability risks regarding HTML Layout Remote Code Execution and URL Validation Vulnerability remain the same in all three releases of Internet Explorer under study. HTML Layout Remote Code Execution has a slightly higher risk than URL Validation Vulnerability.

The second observation is that the security policies adopted by Internet Explorer cannot face sophisticated attacks such as HTML Layout Remote Code Execution, in which case a CVSS score of 7.7 is quite high.

The security policies introduced in IE 8 can decrease the vulnerability score of information disclosure through the XSS Filter.

Yet there is no announced vulnerability on Internet Explorer 10 which is available in platform preview.

These results can contribute to the evaluation of the business impact of XSS vulnerabilities. The browser-dependent risks must be carefully treated since they allow attackers to have end user privileges and to gain control of the applications. The computed scores confirm the OWASP evaluation, and the position of cross site scripting vulnerabilities on their list [7].

5. CONCLUSIONS AND FUTURE WORK

The paper gives an overview of XSS vulnerabilities from a browser point of view. We study the impact of HTML Layout Remote Code Execution, URL Validation Vulnerability and XSS Filter Information Disclosure for three releases of Internet Explorer (7,8,9). We have used the CVSS vulnerability scoring formula in order to measure the impact of these vulnerabilities on security. The obtained results confirm the OWASP analysis, for exploitability and impact.

[2] http://technet.microsoft.com/en-us/security/bulletin/MS10-002

[3] http://technet.microsoft.com/en-us/security/bulletin/MS11-099

[4] http://technet.microsoft.com/en-us/security/bulletin/MS12-010

In our opinion, XSS vulnerabilities should be carefully treated, and eliminated them can improve significantly the security of the application.

As future direction of our study, we intend to study other forms of XSS vulnerabilities, that are difficult to detect with dedicated tools, such the ones due to using ActiveX and Silverlight.

6. REFERENCES

[1] Security, Anti-Cross Site Scripting Library, http://msdn.microsoft.com/en-us/security/aa973814

[2] Jeff Atwood, Coding horror, Protecting Your Cookies: HttpOnly, August 28, 2008

[3] Barracuda Networks, http://www.barracudanetworks.com/ns/products/web-site-firewall-overview.php

[4] Brooks M., Bypassing Internet Explorer's XSS Filter, Traps Of Gold–Defcon 2011, https://sitewat.ch/files/BypassingInternetExplorer'sXSSFilter.pdf

[5] Cisco ACE Web Application Firewall, http://www.cisco.com/en/US/prod/collateral/contnetw/-ps5719/ps9586/data_sheet_c78-458627.html

[6] Citrix NetScaler App Firewall http://www.citrix.com/English/ps2/products/product.asp?-contentID=2312027

[7] Open Web Application Security Project, www.owasp.org

[8] Measuring Software Security Assurance, www.cert.org/archive/pdf/2010research-report-measuring.pdf

[9] P. Mell, K. Scarfone, S. Romanosky, A Complete Guide to the Common Vulnerability Scoring System Version 2.0, http://www.first.org/cvss/cvss-guide.pdf

[10] Klein, A. (2005). DOM Based Cross Site Scripting or XSS of the Third Kind. Web Application Security Consortium Articles, 4. Retrieved from http://www.webappsec.org/projects/articles/071105.shtml

[11] Wassermann, G., Static detection of cross-site scripting vulnerabilities, Proc. of ICSE 2008, pg.171-180

[12] Di Lucca, G.A., Fasolino, A.R., Mastoianni, M., Tramontana, P., Identifying cross site scripting vulnerabilities in Web applications, Proc. WSE 2004, pg. 71-80

[13] Howard M., LeBlanc D., Writing Secure Code, Microsoft Press, 2003

[14] Social Hacking, Recent Facebook XSS Atacks Show Incresing Sophistication, April 21, 2011

[15] Lynch D., XSS is fun!, October 20, 2011, http://davidlynch.org/blog/2011/10/xss-is-fun/