Internet application security

Alen Šimec, Ognjen Staničić
Tehnical Polytechnic in Zagreb/Vrbik 8, 10000 Zagreb, Croatia
alen@tvz.hr, ognjen.stanicic@tvz.hr

Abstract – Due to the rising number of internet application and their users, there is also an increase in the amount of developers developing and perfecting them. Oversights by developers while creating an application can cause countless problems and financial losses. In order to minimize vulnerability, it is necessary to take application design into consideration, assume all potential problems, and eliminate them before launching the application. The most frequent type of attack is XSS or Cross Site Scripting. Various flaws of browsers are displayed. Most of them have been corrected, but are nevertheless a good example of some types of attacks.

I. INTRODUCTION
Cross-site scripting (XSS) is a type of computer vulnerability which is found in internet applications. They enable access to scripts which are run on the client side in the user's browser. Cross-site scripting is a vulnerability which can be exploited by attackers in order to bypass computer access control. A research made by Symantec in 2007. has shown that around 80% of all security oversights are due to XSS1. XSS can cause a wide range of unwanted effects, from slight disturbances to computers, to significant security risks, depending on the sensitivity of data contained on them (Symantec, 2010).

In general, cross-site scripting is a method of attack, which uses code vulnerability of internet applications, making it possible for the attacker to send malicious content to the end user in order to collect certain data contained on the computer.

II. ORIGIN OF XSS
Javascript has introduced various functionalities, which improve content and image appearance in web sites. Shortly after Javascript had been implemented, attackers have discovered the ability to inject a page onto another page, with the intent of collecting data from the user. A page could overtake content from another page and display content that had malicious code in the background. That is the origin of the name cross-site scripting or CSS. It is a simple way of bypassing security obstacles in order to attain data from the user. The abbreviation CSS (Cascading-Style Sheet) is also used for formatting HTML documents, so it was inconvenient to use CSS for cross-site scripting. In order to avoid the ambiguity Mark Slemko started calling it XSS, which eventually became the “official” name for it (Spett, 2009.).

III. ATTACK ON THE USER
The term “cross-site scripting“ is an attacking method which forces an internet application to forward malicious executable code to the user, which is then loaded and executed in the user’s browser. The code can be written in various programming languages such as HTML, Javascript, VBScript, ActiveX, Java, Flash etc. The first XSS attacks were recorded in 1990. The most frequent attacks were on social networks such as Twitter, Facebook, Myspace, Orkut etc. During the last couple of years, XSS attacks were the biggest security vulnerability in internet applications. There are three different types of XSS attacks: Persistent or Stored, Non-persistent or Reflected and DOM-based XSS (Crockford, 2008.).

IV. PERSISTENT OR STORED XSS
Persistent or stored XSS works on the principle of injecting code which collects desired data on the victim's page. In this type of attack, the attacker doesn't need to secure url (internet adress) to the user, because the page requests users to input data. A good example is a forum, or a social network. The attacker sends through a web form a message containing code which is saved to a database on the server. When the user visits a web site which queries the database, the result set contains malicious code, which executes and collects personal data from cookies stored on the user's computer. Cookies can contain information about credit cards, and other confidential data.

Example of an attack:

(Cookie Stealing Code Snippet)

The best way of preventing this type of attack is by escaping any content received from users before it is displayed. Symbols should be translated into HTML code, such as &quot; (for quotation symbol) or &lt; (for <). Escaped code is harmless, because it can’t execute in the end user’s browser.

VI. DOM-based XSS

DOM based XSS doesn’t require an internet server for receiving malicious XSS code. Malicious code uses runtime embedding of attacker data on the client side from the page served by the web server. HTML pages can include user data through an internet interface on the client side in the user’s browser. If the HTML page contains Javascript code which includes a location from an external page, it can contain malicious code. An example of such an attack is the address http://www.secure-page.com/index.html which contains the following code:

```html
<html><title>internet security</title>
<head>
<script>
var pos=document.url.indexOf("name=")+5;
document.write(document.url.substring(pos,document.url.length));</script>
</head>
<body>
  this is a (un)secure page
</body></html>
```

In this example Javascript code includes part of the url in the page, not paying attention to security. The attacker can lure the user to click on a link such as http://www.secure-page.com/index.html?name=<script>alert(document.cookie)</script>, which will enable malicious code.
VII. ANALYSIS AND PREVENTION

The WASC Threat Classification is a cooperative effort to clarify and organize the threats to the security of a website. Creation of an internet page invulnerable to cross-site scripting demands quality equipment and a development team consisting of expert developers and network administrators. One must remember that internet application security is a constant work on security and protection improvement. The attacker often changes goals, and seeks various solutions in order to penetrate the defense to reach the desired data. During internet application development, it is important to keep the code clean and without characters and strings which might be exploited. Removing quotes and putting backslashes when setting up database queries is sometimes not enough. The best way of filtering data is using default-deny expressions which include only desired characters.

Most frequent attacks are via internet browsers. In order to maximize security, latest browser updates should always be installed. Attacks can be stopped by limiting scripts coming from external servers. There are firms which do periodical tests on internet application security by simulating attacks on the system from their servers. If the test shows the system to be flawed and letting attacks through, the firm must upgrade the security system based on the detailed information gathered while testing (Dafydd Stuttard, 2007.).

![Incident frequency by WASC Threat Classification](http://moj.tvz.hr)

Figure 2 - Incident frequency by WASC Threat Classification

VIII. SYSTEM ANALYSIS

By analysing the system, we can notice and correct code writing errors. With the increasing number of internet applications, possibilities of coding errors increase accordingly. We have analysed some internet applications we use at the Polytechnic of Zagreb. The first example is an application which teachers use for publishing material and news related to classes. According to the analysis of [http://moj.tvz.hr](http://moj.tvz.hr) using the tool Acunetix Web Vulnerability Scanner², we have got excellent results.

Test results are displayed in the appendix, figure 3.

Second web application we have analysed is the external page of the Polytechnic of Zagreb, which is used for informing future generations of students about education possibilities and curriculum. The page is written well and the analysis has shown excellent results.

Test results are displayed in the appendix, figure 4.

Third example is the Mipro web application ([www.mipro.hr](http://www.mipro.hr)). Analysis has also shown excellent security results. There are no security vulnerabilities.

Test results are displayed in the appendix, figure 5.

Fourth example is an internet application which is used for renting yachts ([www.bosun.com.hr](http://www.bosun.com.hr)). Results have shown numerous security errors which allow access to certain protected pages. Using the application and leaving data is unsafe. The web application was made using an Open Source CMS, and it is crucial to install additional security updates.

Test results are displayed in the appendix, figure 6.

IX. CONCLUSION

Internet pages today are more complicated than they have ever been containing a large quantity of dynamical content, which is accommodated for the user. Dynamical functionality can lead to a higher sensitivity to cross-site scripting attacks and theft of private user data.

It is not important whether the intrusion has occurred from the outer or the inner network, the sole fact that data on personal computers are unsafe threatens the business of firms which can easily lose the market battle if certain confidential information is stolen. Firms which pay attention to security of their applications and computers have higher chances of survival on the market. Firms should realize that with the trust users give them while giving their private data, comes a big responsibility. An example of such an attack happened to the firm Sony Playstation. Users' credit card information was stolen, and the firm not only lost money, but all of their credibility with its users as well. It is assumed that 77 million of user data was lost. That attack is one of the biggest attacks on systems so far, and it really stresses the fact that firms have to pay a great deal of attention to data security.

² [http://www.acunetix.com/vulnerability-scanner](http://www.acunetix.com/vulnerability-scanner)
X. REFERENCES  


XI. Appendix

Figure 3 – Polytechnic of Zagreb internal web application test results

Figure 4 – Polytechnic of Zagreb external web application test results
Figure 5 – Mipro conference web application test results

Figure 6 – Yacht renting web application test results