Introduction to Cross Site Scripting using WebGoat

The OWASP LiveCD Education Project

Author: Brian Blankenship

SECUR[IT]Y DISTRO

www.securitydistro.com

Table of Contents

A1 Abstract

Cross Site Scripting is one of the most common web vulnerabilities in existence today, and subsequently one of the most exploited issues.

This tutorial is geared towards someone who may have heard of cross site scripting, and may even understand the concepts behind it, but would benefit by seeing real examples and having the opportunity to experiment with them.

A2 Prerequisites

WebGoat – A deliberately insecure application maintained by OWASP. There are several ways you can setup WebGoat which will be outlined later in this document.

Web Browser – FireFox was used for this tutorial, but most any web browser should work.

.

A3 Objective

Setup a working WebGoat installation, gain a understanding of what cross site scripting is and how it works, and work through some basic cross site scripting attacks.

A4 Background on Cross Site Scripting

The term “Cross Site Scripting” can be a bit confusing as it might imply some sort of script that is used for evil purposes across multiple areas of a web site. To add further to the confusion, it started off being referred to as “CSS” which also stands for “Cascading Syle Sheets”. Now days it is most commonly referred to as “XSS”, or simply spelled out completely as Cross Site Scripting.

So what is XSS? It is simply tricking a web server into delivering malicious content to another user. The server is the delivery mechanism, but the malicious code runs in the victims web browser.

XSS attacks are divided into two main categories; reflected and stored. A third type called DOM Based XSS exists but is out of the scope of this tutorial. If you wish to read about DOM Based attacks, check out the paper written by Amit Klein at
http://www.webappsec.org/projects/articles/071105.html

A reflected XSS (non-persistent) attack is one that uses a separate mechanism such as a second web server, an email, or some other delivery mechanism. The effect is the same, but the attack is interactive. For example, a person sends an email with a link to a well known web site. The link seems harmless because it goes to a known site, but the link also contains extra code that runs a malicious script from another site. The URL can be encoded also, obscuring the malicious portion which makes it difficult to identify.

A stored XSS (also referred to as “persistent”) attack works like this. A person stores content with embedded malicious code on a web page, in a guest book, or any other permanent place on a server. Unsuspecting users later view that content and the malicious code executes in their browser. This makes stored attacks more dangerous because even a XSS savvy person would have no way of anticipating the attack until it is too late.

In the earlier days of the Internet, most web pages were static. A user could view a web page, but could not add to it or modify the content. Today lots of web pages have dynamic content, many of which are vulnerable to stored XSS attacks.

Ok, now that we have a basic understanding of what XSS is, let's fire up WebGoat and work through a couple live examples.

A5 WebGoat Setup

WebGoat can be setup and used in multiple environments including:

- **OWASP Live CD (LabRat)** – LabRat is a compliation of security tools that includes WebGoat. The LabRat CD is a downloadable .ISO image that once burned to a CD can run on any PC that can boot from CD. It uses a “live” Linux distribution that does not affect the operating system installed on your hard drive. You can use WebGoat and the entire collection of LabRat tools without installing anything. The LabRat .ISO file can also be run under VMware if you prefer.
- **Linux Installation**
- **Windows Installation**

For this tutorial a Windows installation of WebGoat was used, but feel free to use whatever method works best for you. If you decide to use the Live CD, VMware, or a Linux installation, just skip past the next session and go directly to the examples.

Links to all of the WebGoat versions and the OWASP Live CD can be found under the download section at http://owasp.org

Windows WebGoat Installation:

To get Webgoat, visit http://owasp.org and go to the downloads section. Select the link for WebGoat, then the link for “OWASP Source Code Center at Sourceforge” to get to the download area for the Windows version of WebGoat.

Download Windows_WebGoat-5.0_Release.zip and save it to your local drive. Double-click the .zip file and copy the WebGoat-5.0 folder to wherever you like on your system.

Before launching WebGoat, please review the readme.txt file included with it. It is important to understand that WebGoat creates an intentionally insecure web site on your system that could be used to attack you if your system is on a network without a firewall between you and other users. Also mentioned in the readme file is the importance of limiting security testing to only systems that you own, or have permission to work with. A person can quickly get themselves into trouble testing the security of other systems even if their intentions are good.

Now that we have that out of the way, let's begin by starting WebGoat. Navigate to the WebGoat-5.0 directory where there are two batch files you can launch it with. Wwebgoat.bat will start it using port 80, and webgoat_8080.bat will do the same thing but the web server will listen on port 8080.

WebGoat has a Tomcat web server built in that requires no configuration, making it really easy to get a test system up with minimal effort.

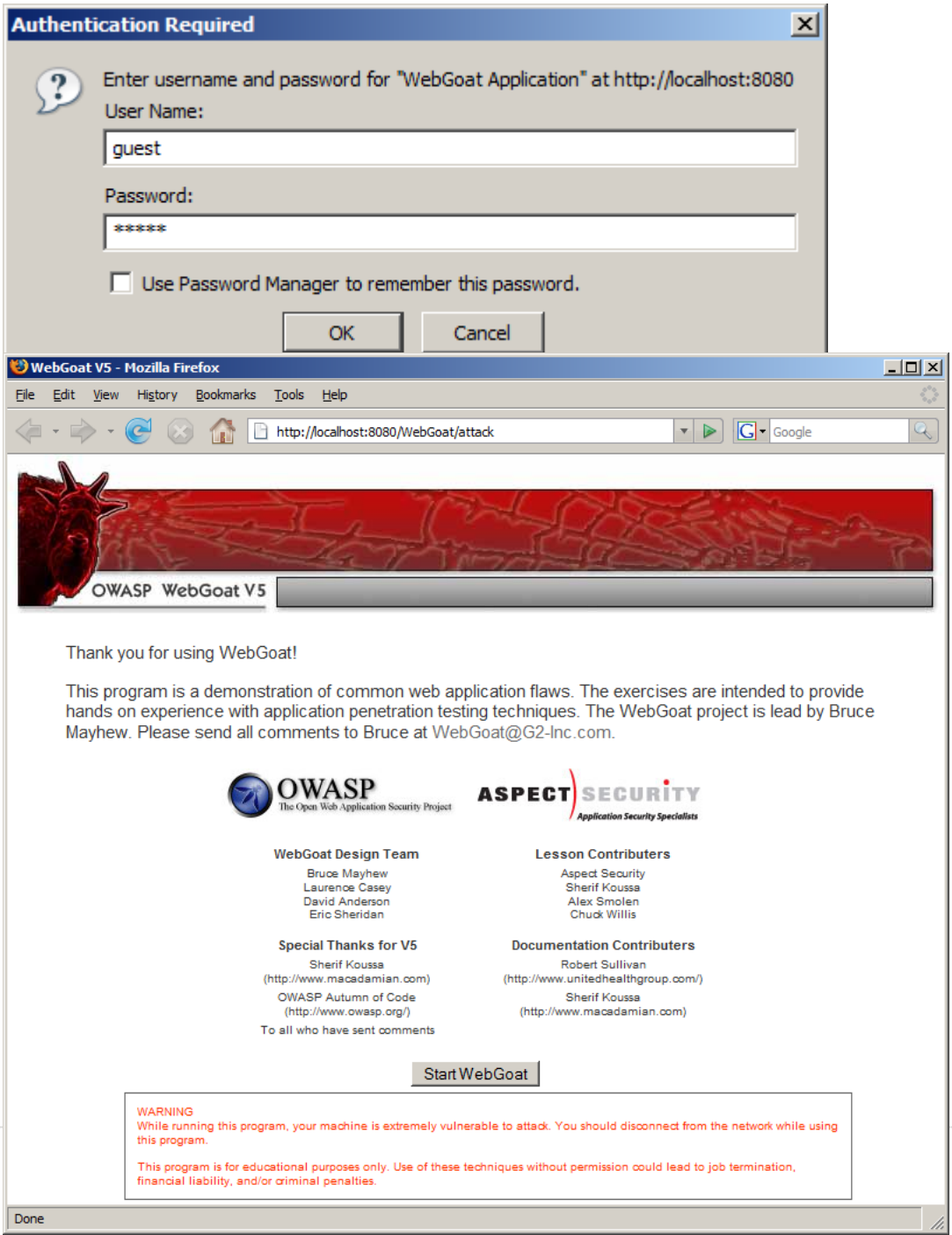
Launch the webgoat_8080.bat file by double-clicking on it.

You may receive a message from the Windows firewall asking if you want to allow communications or keep blocking, select allow.

Now startup your browser and enter http://localhost:8080/WebGoat/attack

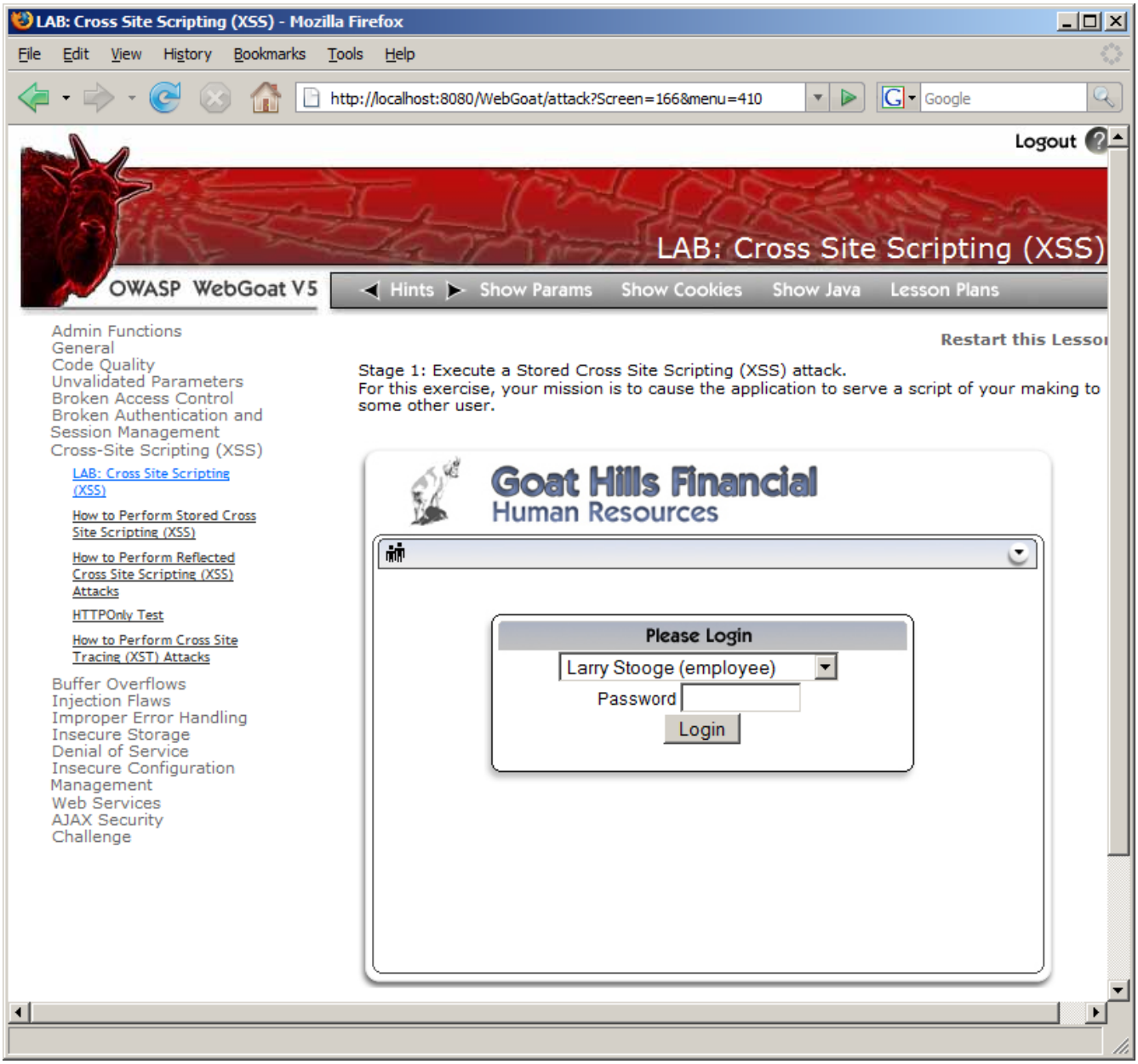
Note: The URL is case sensitive.

You will be prompted to login. Enter “guest” for the username and the password.



You should now be a the main WebGoat web page. Click the “Start WebGoat” button.

When the next page comes up, click on “Cross Site Scripting (XSS) on the left side to get to expand the XSS section of WebGoat.



A6 Examples of XSS attacks

Let's try a reflected XSS attack.... Click on the link “How to Perform Reflected Cross Site Scripting (XSS) Attacks”.

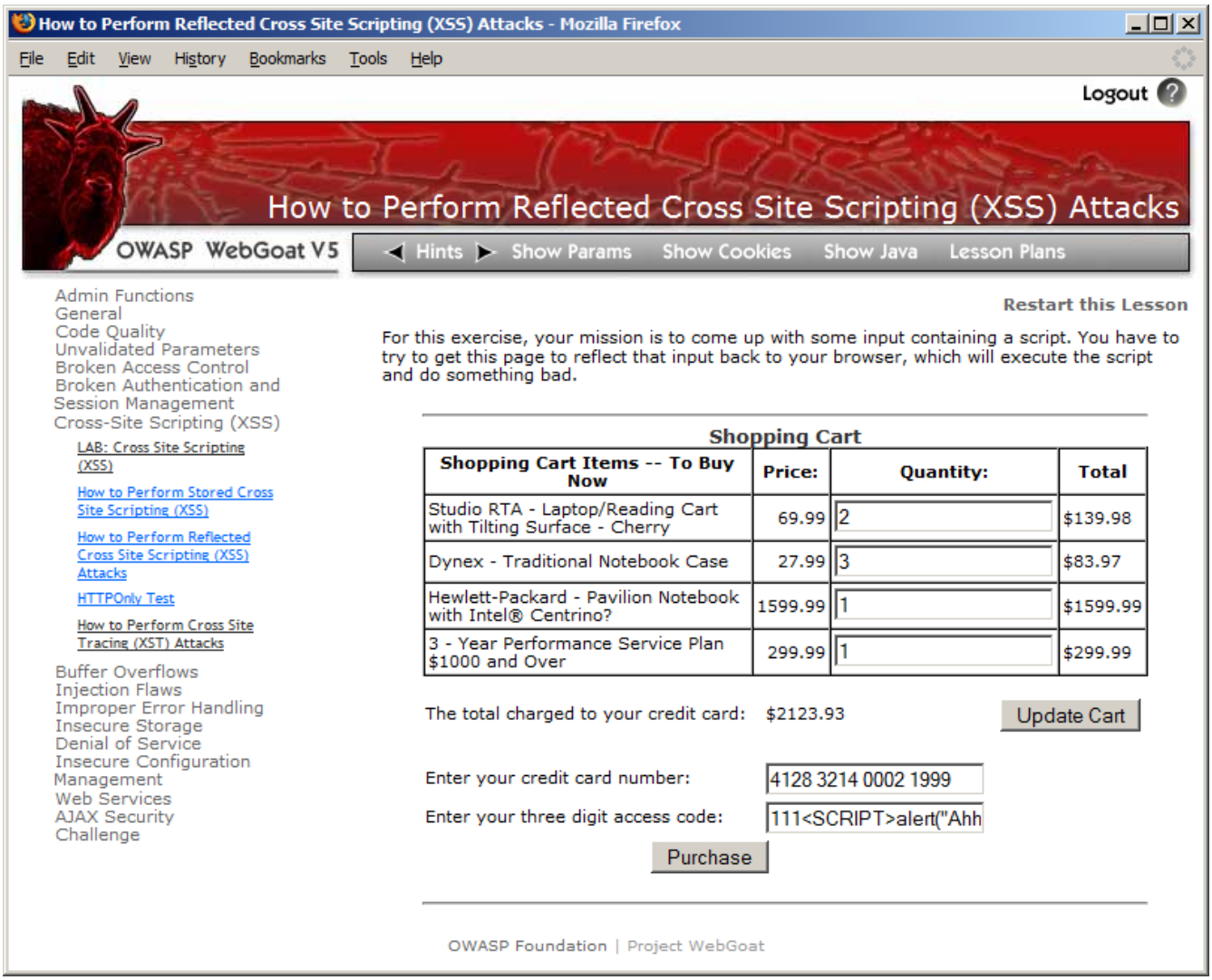
This page on WebGoat page resembles a typical shopping cart checkout page. The inputs to this page include the quantities for the items selected, the credit card number, and the three digit access code for the credit card. You can change quantities and click on update and the page will update the prices.

As you might have guessed, this page has a XSS vulnerability. The three digit access code field does not have any input validation. In a real world application like this, we would want to only allow numeric digits to be entered into the access code field, and limit the length to three.

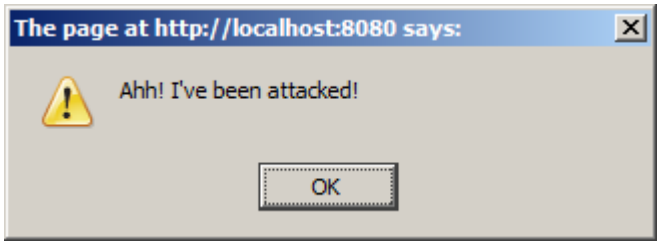
Remember that we are tring to perform a reflected attack, so we want to provide input to the server that includes some extra content that will be sent back to your browser when it redisplays the page.

Try adding the following script to the end of the access code field, and click “Purchase”.

<SCRIPT>alert("Ahh! I've been attacked!")</SCRIPT>



If it worked correctly, you should see the message below displayed.



So what happened here? When you clicked the “Purchase” button, the data input into the form was submitted to the web server. The server side code did not perform any validation on the access code field, allowing the script code to be accepted along with the “111” access code. When the server sent the updated information back, the script was included which executed in your browser.

While this is entertaining, you might be wondering how this type of attack can be used. Obviously people are not going to accidentally attack themsleves with script code, but what if someone sends you a link to a web page that has a script embedded into the URL? A malicious URL can be sent out to thousands of people via SPAM, Instant Messaging, or mechanisms. While this example is harmless, there are endless possibilities of what can be done.

Example of a Stored XSS attack:

Stored XSS attacks can be more dangerous for several reasons. First off, it is easier to get someone to run it. When you receive unsolicited email, you probably don't click on the links they may contain. But what if you are simply reading messages on a forum you visit regularly? Embedded scripts can launch automatically when the message is displayed.

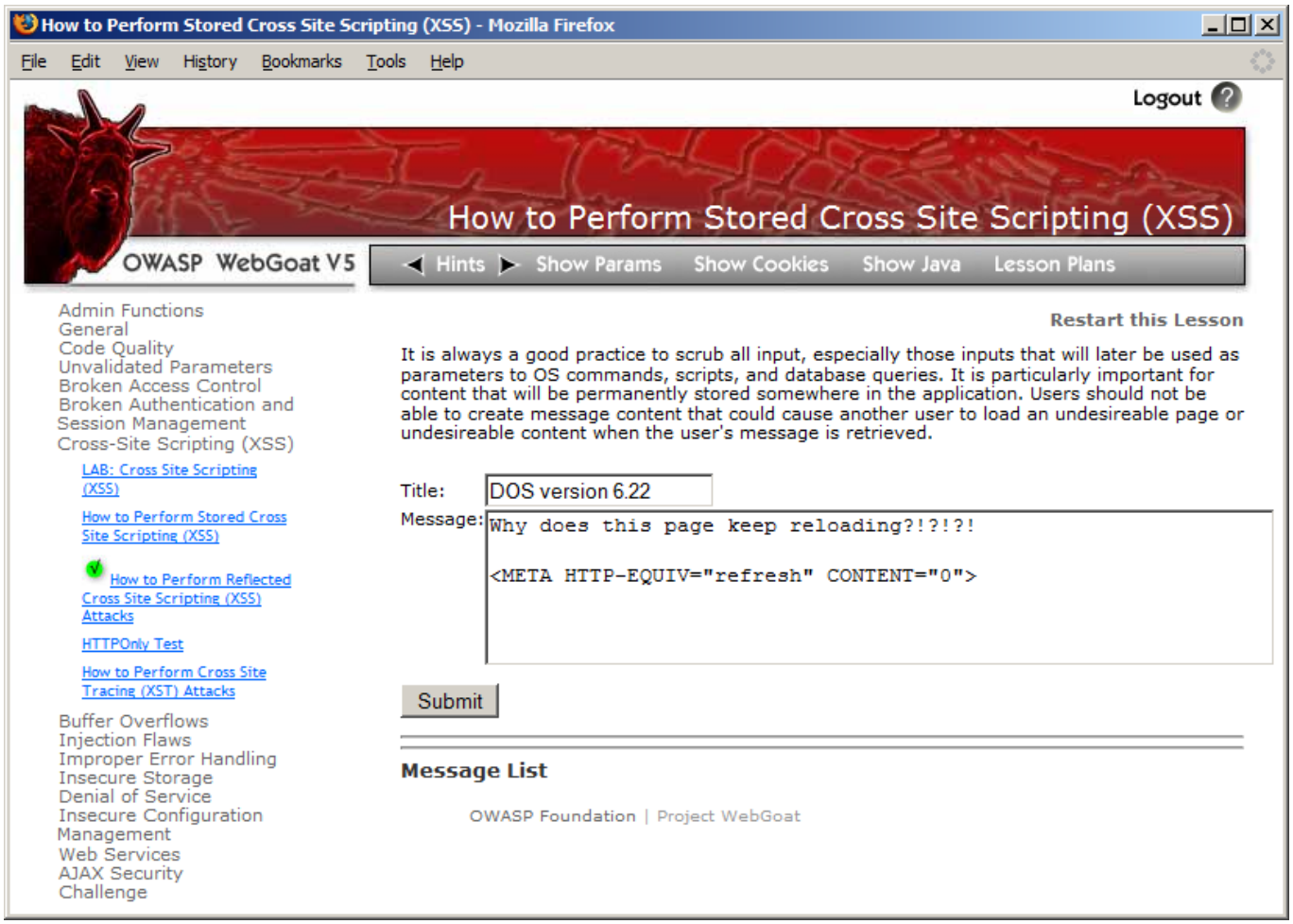
To make matters worse, you may be logged into a site when your browser executes the malicious code. This makes it much easier for a script to steal cookies, allowing the attacker to hijack your session. Again, the possibilities are endless as to what can be done.

Now let's try an example of a stored XSS attack. This attack would primarily be annoying for the recipients, but it could create a DOS (Denial of Service) condition if enough people viewed the content simultaneously. Click on the “How to Perform Stored Cross Site Scripting (XSS)” link.

This page of WebGoat simulates a message board. You can enter a title and message text, click Save, and it will be listed in the Message List at the bottom of the page. Enter a message like the one in the screenshot below and then click “Submit”.

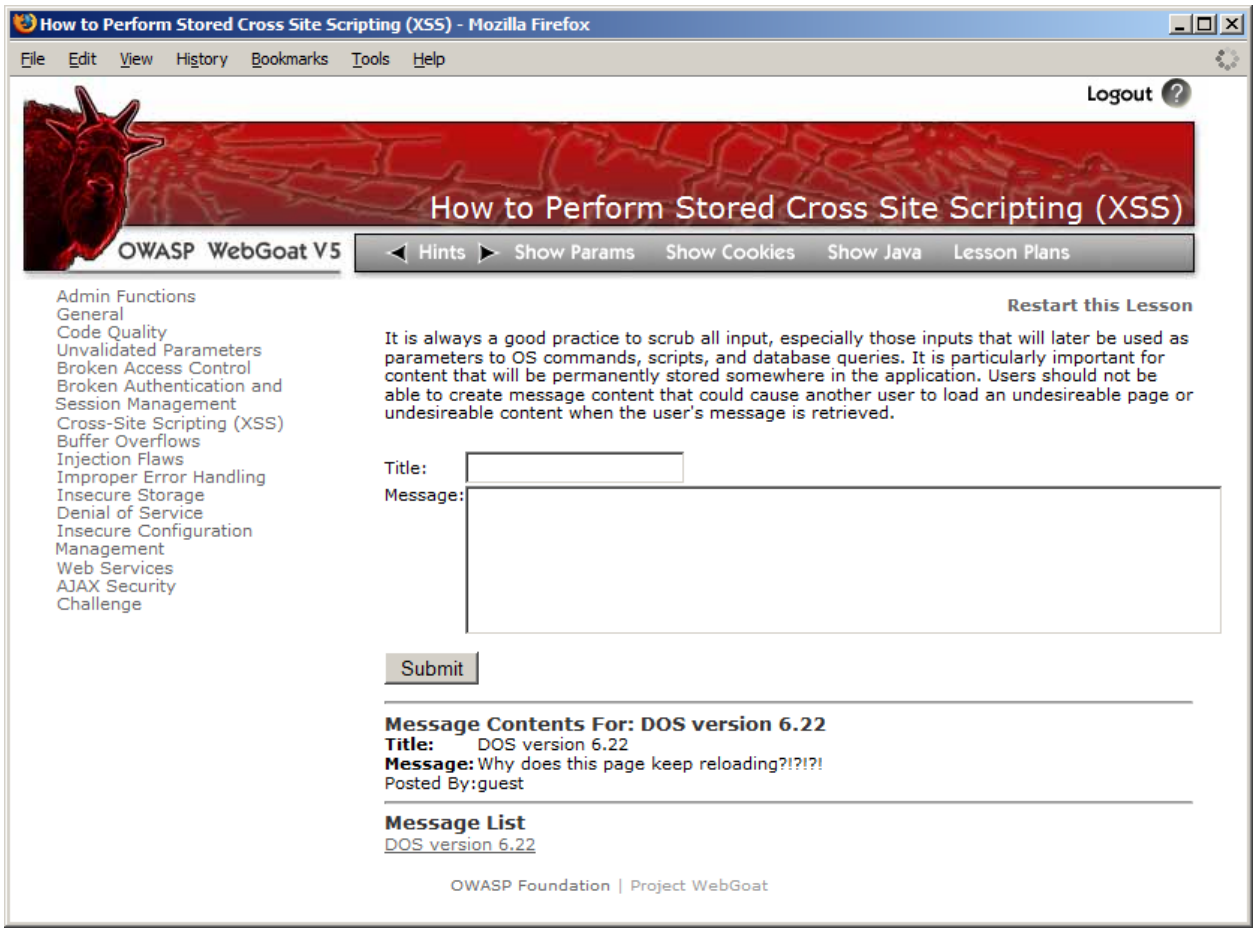
The message title and content are not important, only the script portion:

<META HTTP-EQUIV="refresh" CONTENT="0">



After the page refreshes, your message title will be displayed in “Message List” at the bottom of the page. Click on it to see the results.

You should see a page like the one below. If you successfully entered the HTML portion of the message, it should be refreshing repeatedly in your browser. Just hit your browsers stop button when it becomes annoying.



Notice that the HTML portion of the message is intercepted by your browser and does not get displayed.

This concludes the examples for this tutorial. Hopefully you have gained a basic understanding of how stored and reflected XSS attacks work. Now that you have a functional WebGoat installation you may want to explore the other lab examples included with it.

A7 Conclusion

This concludes the examples for this tutorial. Hopefully you have gained a basic understanding of how stored and reflected XSS attacks work. Now that you have a functional WebGoat installation you may want to explore the other lab examples included with it.

A8 About the Author

Brian Blankenship works as a Consulting Security Analyst focusing on vulnerability management for Kindred Healthcare in Louisville, KY. He has twenty years of experience with information systems and maintains CISSP and SANS GSEC certifications.