WebScarab – Exploiting Input Validation through Parameter Tampering

Author: Abraham M. Smith

SECUR[IT]Y DISTRO

www.securitydistro.com

Table of Contents

A1 Introduction

One of the fundamental security vulnerabilities that exist among many of the popular flavors of exploits such as XSS and SQL Injection as well as many others is the lack of input validation. Web applications that check if client input is valid and useable by the application for which it was designed, have taken the first and probably the most effective step in securing their web application. We will attempt to familiarize the reader in this tutorial on how to tamper with and exploit parameters that could lack of input validation.

A2 Prerequisites

In order to follow the steps in this tutorial you need to have the following.

WebScarab - Either through LabRat or by downloading the java version.
http://dawes.za.net/rogan/webscarab/WebScarab.jnlp

WebGoat - Either through LabRat or by configuration on your local host.
http://sourceforge.net/project/showfiles.php?group_id=64424&package_id=61824

A3 Objective

In this tutorial we will be examining and testing for input validation through parameter tampering using WebScarab, as well as familiarizing ourselves with the user interface.

If you are completely new to WebScarab, it is recommended that you read the WebScarab Getting Started Guide before completing this tutorial.

http://www.owasp.org/index.php/WebScarab_Getting_Started

Three locations in web application that are examples of places that attacks on parameters might exist are:

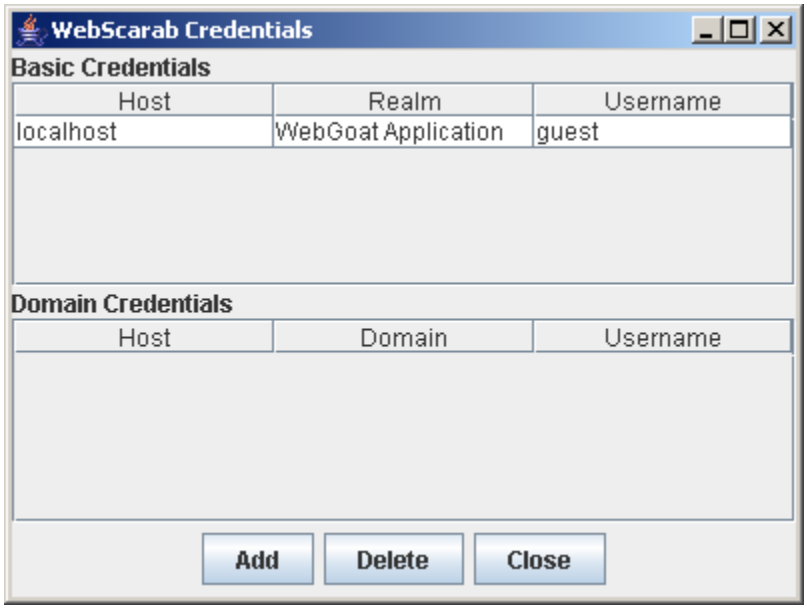
- Hidden Fields
- URL Query Strings
- Cookies

Parameter use is not limited to these three areas; they are examples or parameters that are subject to tampering. In this tutorial, we will focus on field values.

In this WebScarab tutorial, we will be using WebGoat as the target web application for working examples. Before launching Web Scarab make sure WebGoat is up and running. If you are not familiar with WebGoat or how to get it running please refer to (link). Once you have confirmed that WebGoat is up and running you need to launch WebScarab and then your browser. To configure your browser refer to the WebScarab Getting Started Guide (link).

A4 Form Fields – Displayed and Hidden

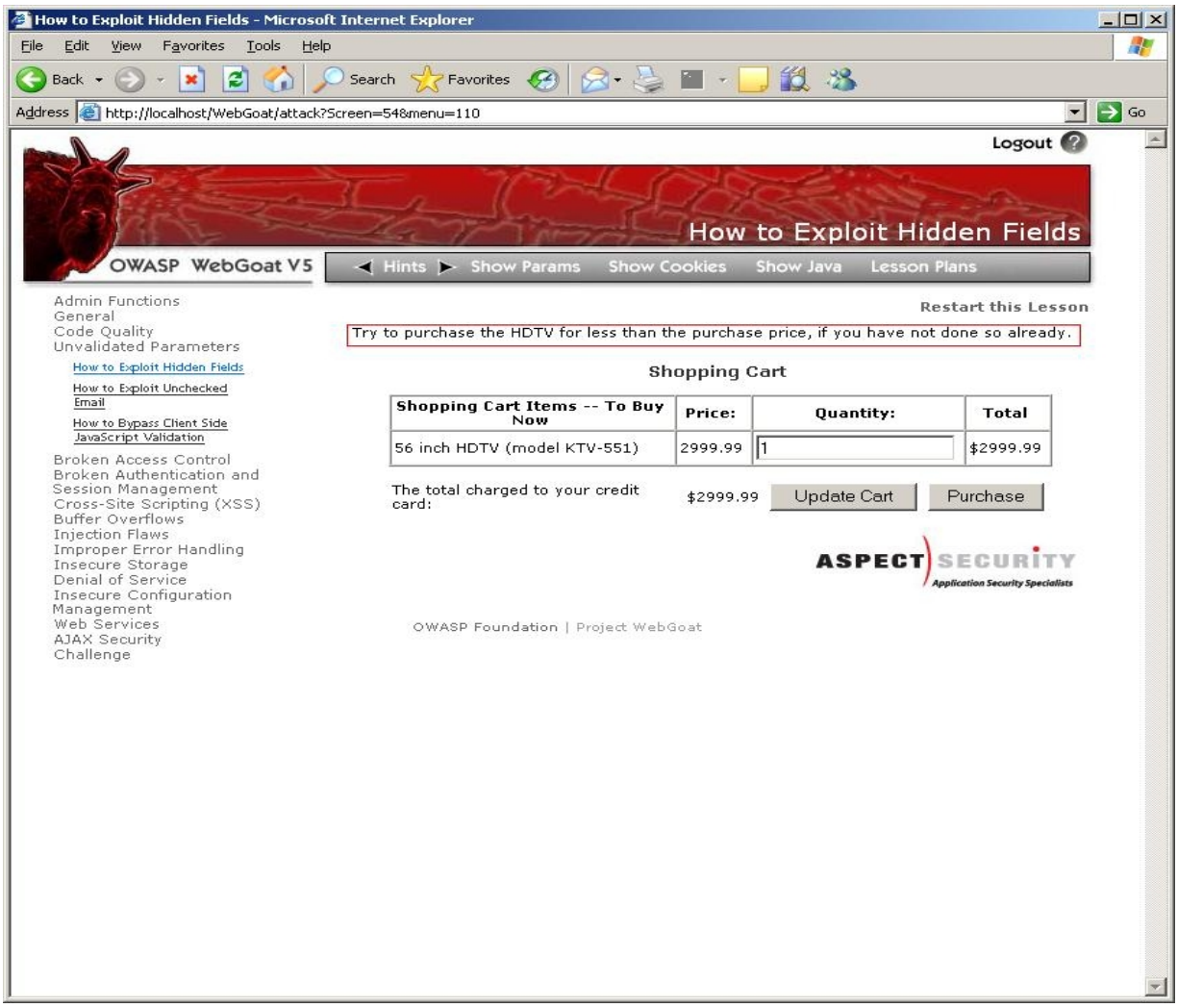
1. In you browser load the WebGoat home page and logon. http://localhost/WebGoat/attack (do not forget to capitalize the W and G, you may also be prompted by WebScarab for you credentials. These credentials will be kept by WebScarab and used should the application prompt for there use again. You can also enter you credentials prior to starting WebScarab's proxy. Tools>Credentials>Add)



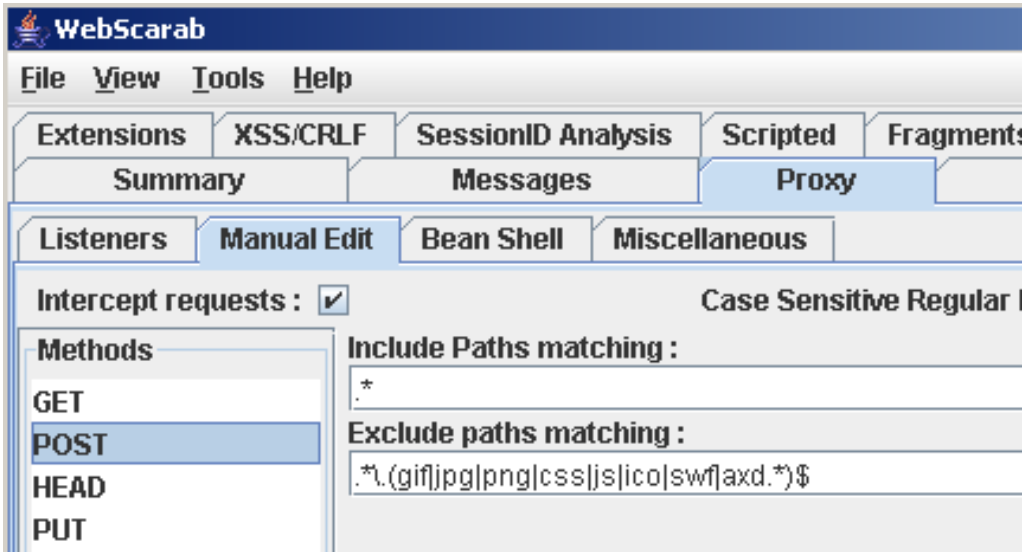
2. In your browser with WebGoat loaded click on Start.

3. On the right menu click on "Unvalidated Parameters" and then "How to Exploit Hidden Fields."

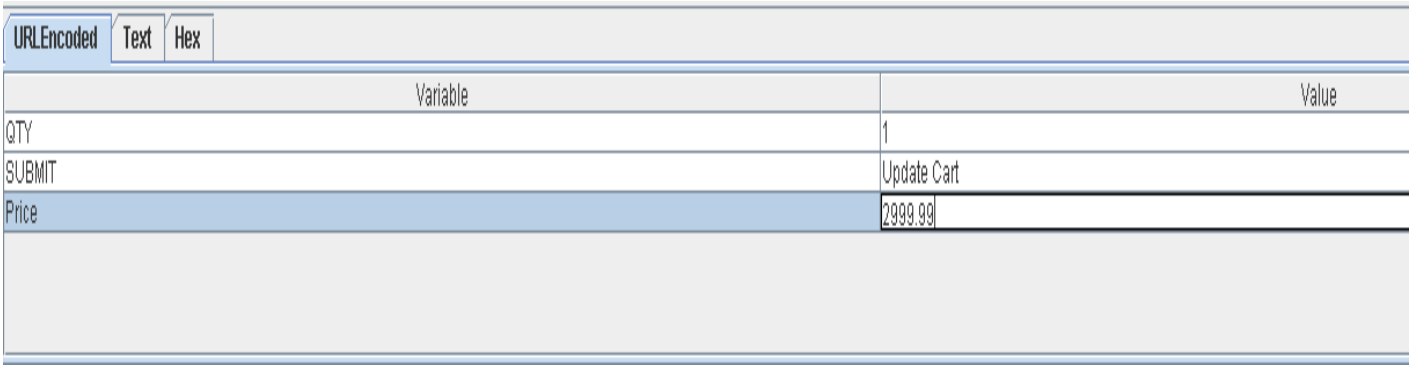
You will notice at the top of the page a brief description of what the lesson would like for you to attempt to do. Change the price of the TV by tampering with the hidden filed parameter. Let's identify the hidden parameter from within the post request and then look at a feature in WebScarab that makes identifying hidden fields on a page easy.



4. In WebScarab go, Proxy>Manual Edit tab. Select the checkbox to intercept requests and highlight the post method. This will setting will intersect any post request method made to the web application and display it for the user to manipulate or examine before sending it on the web application.



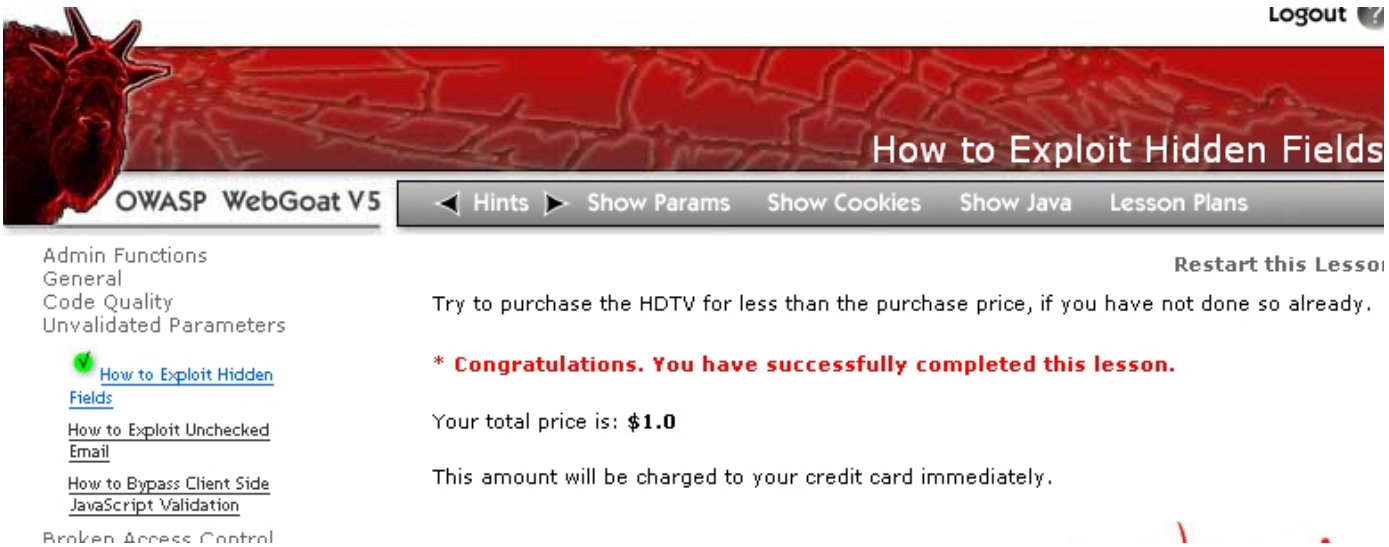
5. Go back to your browser with the "How to Exploit Hidden fields" page displayed and click on "Purchase" this will send a post request to the web server and allow WebScarab to intercept the request to be displayed. You should get a popup from WebScarab.



You will see that WebScarab parses the request into an easy to read format and breaks down the information into logical groups. Listed in one window are the variables for the POST and their values. Spend some time looking at this request and toggle between the raw and parsed view to see how the request is parsed out into logical areas.

6. Edit the variable Price. Change it from 2999.99 to 1.00. Notice how this value changes in the text, hex and raw section of the request. Click "Accept Changes" at the bottom of the request.

Go back to your browser and you will notice you have successfully change the price of the TV by manipulating the price parameter value in the request.



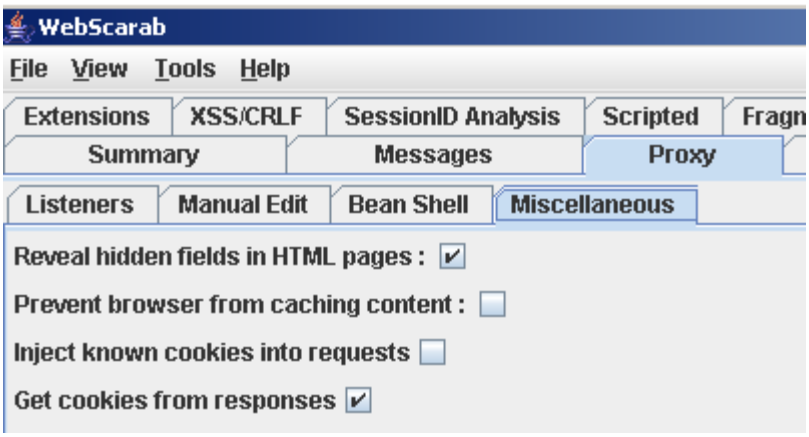
Looking at it from the raw view of the header from the post, all we have done is changes the following highlighted value.

POST http://localhost:80/WebGoat/attack?menu=110 HTTP/1.1
*Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/xaml+xml, application/vnd.ms-xpsdocument, application/x-ms-xbap, application/x-ms-application, */**
Referer: http://localhost/WebGoat/attack?Screen=54&menu=110
Accept-Language: en-us
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; Avant Browser; .NET CLR 1.1.4322; .NET CLR 2.0.50727; InfoPath.1; .NET CLR 3.0.04506.30)
Host: localhost
Content-length: 35
Proxy-Connection: Keep-Alive
Pragma: no-cache
Cookie: JSESSIONID=549E61B035FF33EA06DF6FB58E00911E

QTY=1&SUBMIT=Purchase&Price=2999.99

That does not mean that the web application will automatically accept that value as the price of the TV, especially if the web application performs input validation. All we have done is examined how to manipulate a post request using WebScarab. Next, we will look at how to tamper with a hidden value in response from the web server. Note: this method of tampering with the server data as it is being sent to the client browser is one of the possible ways to defeat client side controls that rely on values from the a server response as a form of validation. If a web server is only looking for possible tampering of a POST method then tampering with the response of a GET request could be the weakness.

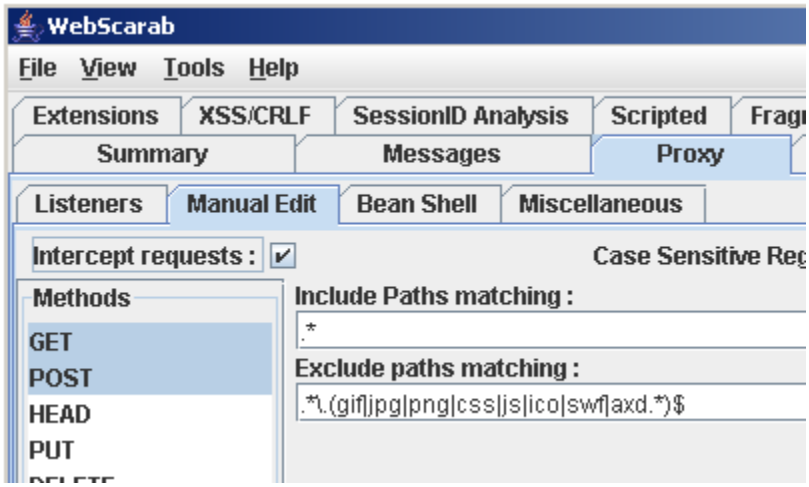
7. In WebScarab go to Proxy>Miscellaneous and select “Revel Hidden Fields in HTML pages”



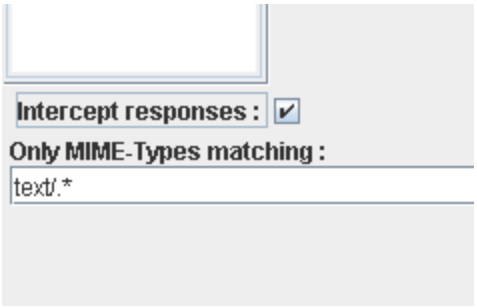
8. In WebScarab go to Proxy>Manual Edit and clear the check box to intercept requests.

9. In your browser click on "How to Exploit Hidden Fields" again to refresh the page. Notice how the hidden field is displayed with the price of the TV.

10. Now, let's go back to WebScarab and set the proxy to intercept both GET and POST requests. (Hold down CNTRL key while clicking to select multiple methods)



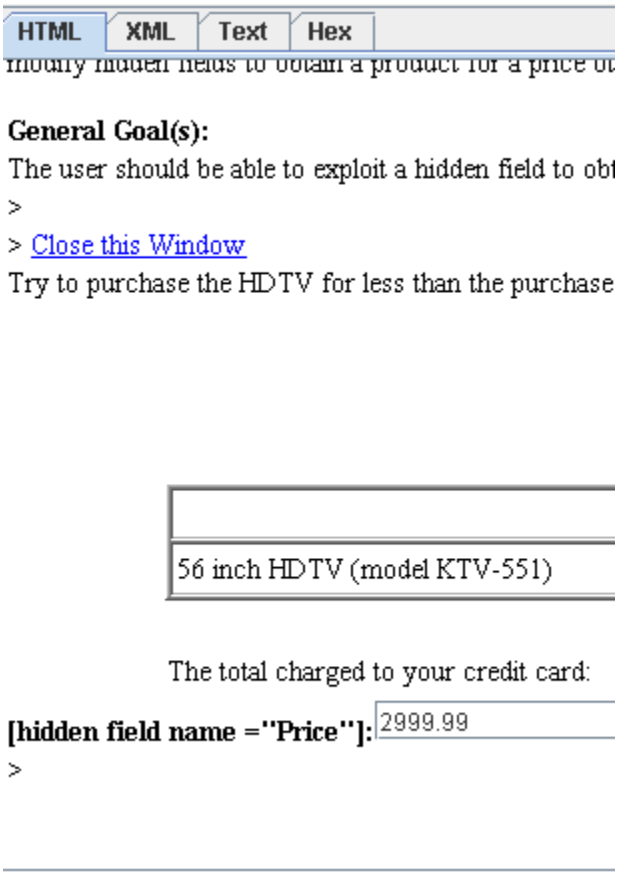
11. On the same page set the proxy to "Intercept responses:" This will allow you to tamper with the data coming from the server before your browser uses it.



12. Now back to the browser and refresh the page by clicking on the link “How to Exploit Hidden Fields” WebScarab will display the GET request for the link.

13. Click “Accept Changes”, now WebScarab shows the response from the server.

14. Click on the HTML tab and scroll to the bottom, there you will see the hidden field being sent back by the web server. Note the field name.



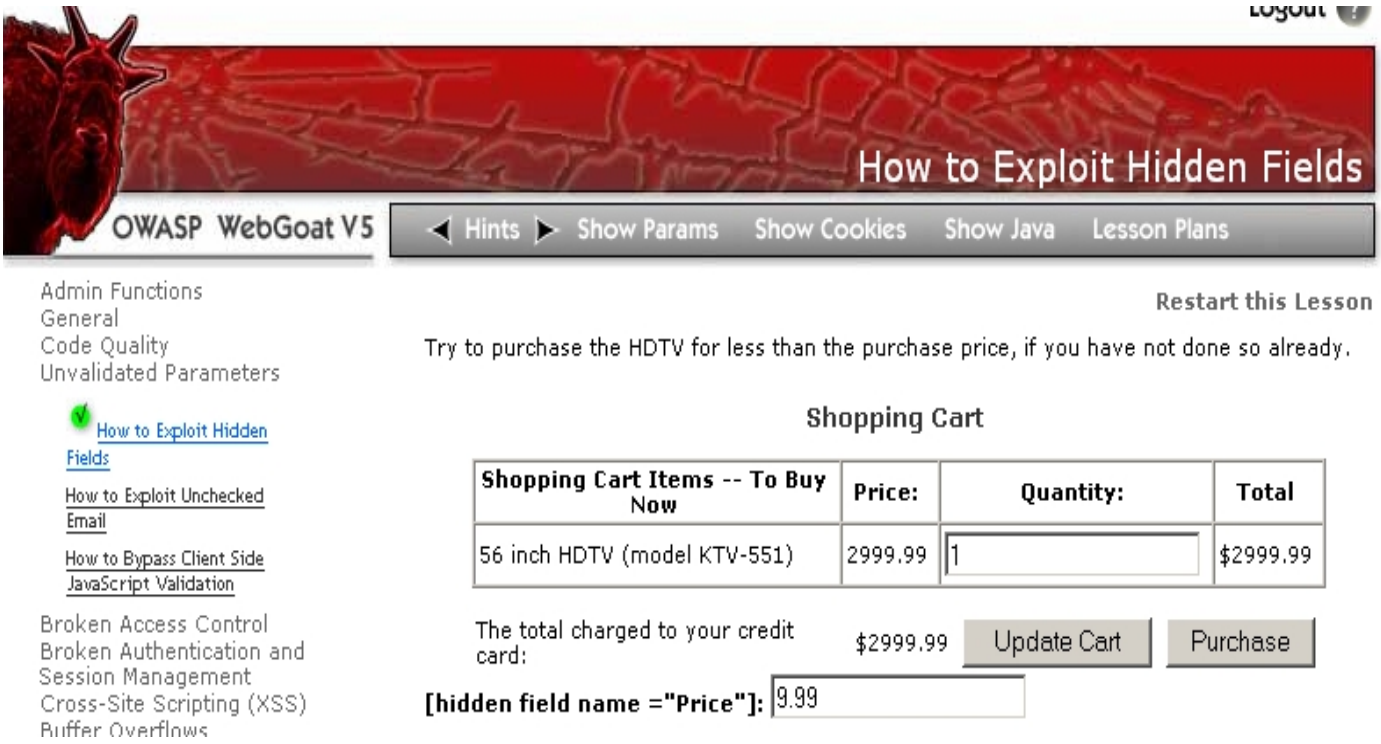
15. Select the Text tab and find the hidden field and change the value from 2999.99 to 9.99. (Scroll to the bottom and work you way up)

*</table><STRONG style="background-color: white;"> [hidden field name ="Price"]:</STRONG> <input name='Price' type='text' value='9.99'>

*

16. Click on “Accept Changes”. At this point you can either accept all intercepts or deselect “Intercept Requests” and “Intercept Response” at the top of the edit window and click on “Cancel all Request Intercepts” depending on if you want to look at every request response with the web server.

17. Go back to your browser when WebScarab is complete and you should now see that the hidden filed has been modified with the amount you placed in the response from the server.



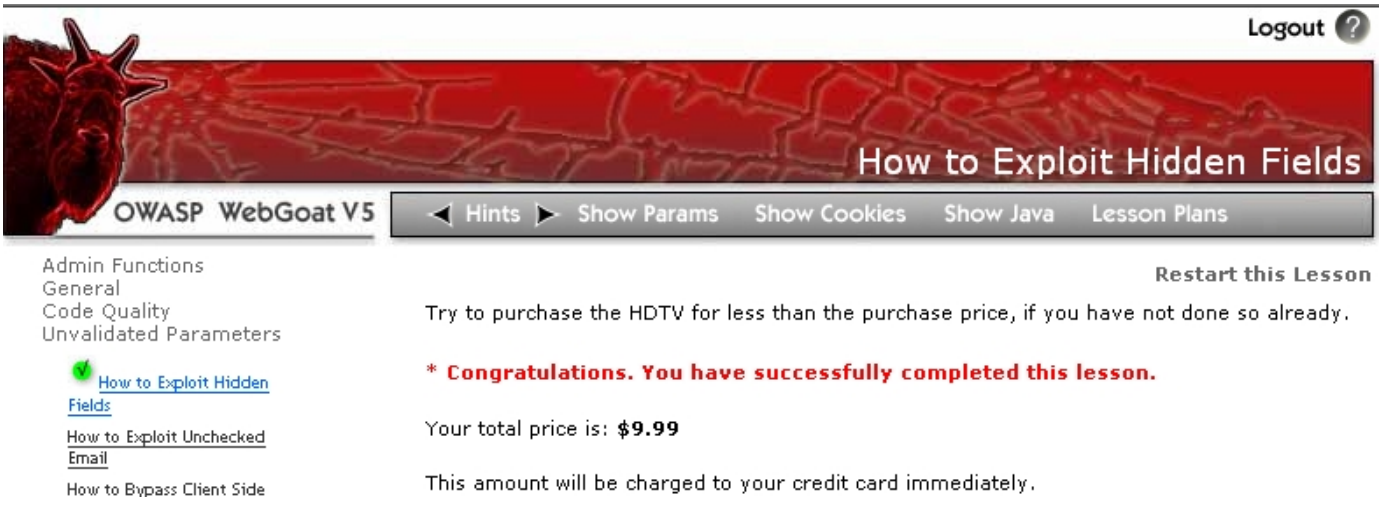
18. Now that you have already manipulated the field you can just click on Purchase to order your TV for 9.99. Before you do, set WebScarab to intercept the conversion with the web server to see just how the data is being sent to the server from the hidden field.

POST http://localhost:80/WebGoat/attack?menu=110 HTTP/1.1
*Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/xaml+xml, application/vnd.ms-xpsdocument, application/x-ms-xbap, application/x-ms-application, */**
Referer: http://localhost/WebGoat/attack?Screen=54&menu=110
Accept-Language: en-us
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; Avant Browser; .NET CLR 1.1.4322; .NET CLR 2.0.50727; InfoPath.1; .NET CLR 3.0.04506.30)
Host: localhost
Content-length: 32
Proxy-Connection: Keep-Alive
Pragma: no-cache
Cookie: JSESSIONID=549E61B035FF33EA06DF6FB58E00911E

QTY=1&SUBMIT=Purchase&Price=9.99

You will notice the value from the hidden field has been placed in the POST request. ???????

19. Clear the intercept options and select cancel all intercepts. Return to your browser to see the response from the server.



End of Tutorial

A5 Conclusion

This tutorial only scratches at the surface of one aspect of web application security. I hope that you

have gained some insight into parameter tampering and see the value of input validation as a fundamental security practice for any web application.

A6 About the Author

Abraham Smith is a CISSP with 9 years of technical experience with a focus on information security over the last 6 years.