Web attacks and security: SQL injection and cross-site scripting (XSS)

License

Contents

General notes about the labs

Often the lab instructions are intentionally open ended, and you will have to figure some things out for yourselves. This module is designed to be challenging, as well as fun!

However, we aim to provide a well planned and fluent experience. If you notice any mistakes in the lab instructions or you feel some important information is missing, please feel free to add a comment to the document by highlighting the text and click the comment icon ( ), and I (Cliffe) will try to address any issues. Note that your comments are public.

If you notice others are also reading the lab document, you can click the chat icon ( ) to discuss the lab with each other.

Preparation

As with all of the labs in this module, start by loading the latest version of the LinuxZ template from the IMS system. If you have access to this lab sheet, you can read ahead while you wait for the image to load.

> To load the image: press F12 during startup (on the blue boot screen) to access the IMS system, then login to IMS using your university password. Load the template image: LinuxZ.

Once your LinuxZ image has loaded, log in using the username and password allocated to you by your tutor.

The root password -- **which should NOT be used to log in graphically** -- is "tiaspbiqe2r" (**t**his **i**s **a** **s**ecure **p**assword **b**ut **i**s **q**uite **e**asy **2** **r**emember). Again, never log in to the desktop environment using the root account -- that is bad practice, and should always be avoided.

Using the VM download script (as described in the previous lab), download and **start** this VM:

- Web Security Dojo

Feel free to read ahead while the VMs are downloading.

Introduction to web attacks and security

Over time there has been a shift to writing and hosting software via the World Wide Web (WWW), and while software vulnerabilities are discovered daily, the security of critical infrastructure such as operating systems themselves is generally improving. Attacks targeting client-side applications and other services are major sources of threat. With the move to Web-based software comes a plethora of new categories of vulnerabilities, such as programming errors common in Web apps.

Many Web apps are written by programmers who are not aware of the security issues, and consequently in many cases organisations develop bespoke software that is vulnerable to attack. These flaws often allow attackers to access or modify databases or attack other users of the website.

Understanding Web traffic

At its most basic, HTTP is a simple protocol that involves a client (for example a Web browser, such as Firefox or Chromium) making requests for web pages from the server (such as Apache or IIS).

This can be visualised, by talking to a Web server using a simple TCP tool such as Netcat.

Open a terminal ("Applications" menu, "Accessories", "Terminal"), and run:

```
nc localhost 80
```

The Netcat process will connect to the Apache server running locally, and now you can type in commands to send to the server. Type:

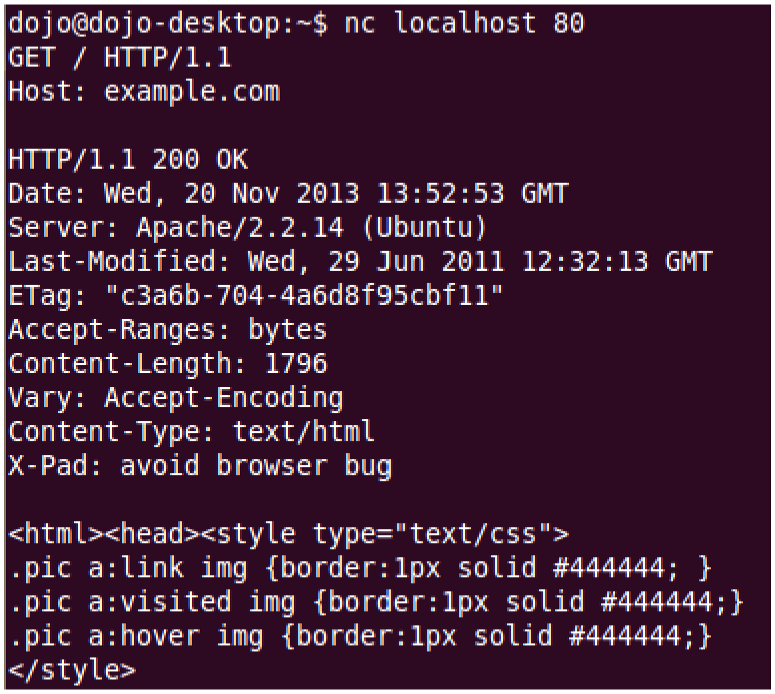
```
GET / HTTP/1.1

Host: example.com
```

Press the Return key twice.

Tip: after starting Netcat you will need to type the above quickly, or copy and paste.

The server will respond with the HTML for a webpage. This is exactly how a Web browser works, except that the browser would take the HTML code and use it to render and display the page graphically.



```
dojo@dojo-desktop:~$ nc localhost 80
GET / HTTP/1.1
Host: example.com

HTTP/1.1 200 OK
Date: Wed, 20 Nov 2013 13:52:53 GMT
Server: Apache/2.2.14 (Ubuntu)
Last-Modified: Wed, 29 Jun 2011 12:32:13 GMT
ETag: "c3a6b-704-4a6d8f95cbf11"
Accept-Ranges: bytes
Content-Length: 1796
Vary: Accept-Encoding
Content-Type: text/html
X-Pad: avoid browser bug

<html><head><style type="text/css">
.pic a:link img {border:1px solid #444444; }
.pic a:visited img {border:1px solid #444444;}
.pic a:hover img {border:1px solid #444444;}
</style>
```

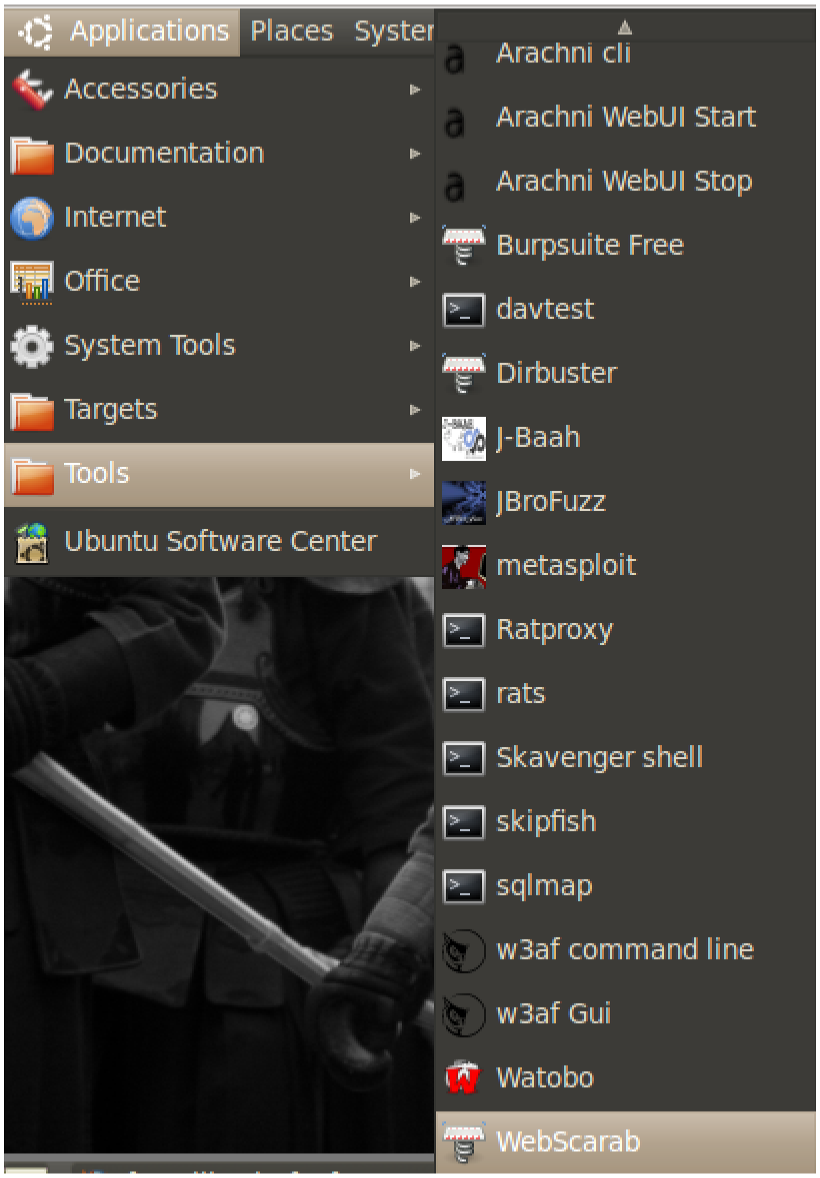
Simulating the job of a Web browser using Netcat

Read through the response.

Browse to the same page using Firefox http://localhost, and right-click the page and "View Page Source". Confirm this is the same source you obtained manually.

Using a local Web proxy to intercept and modify Web traffic

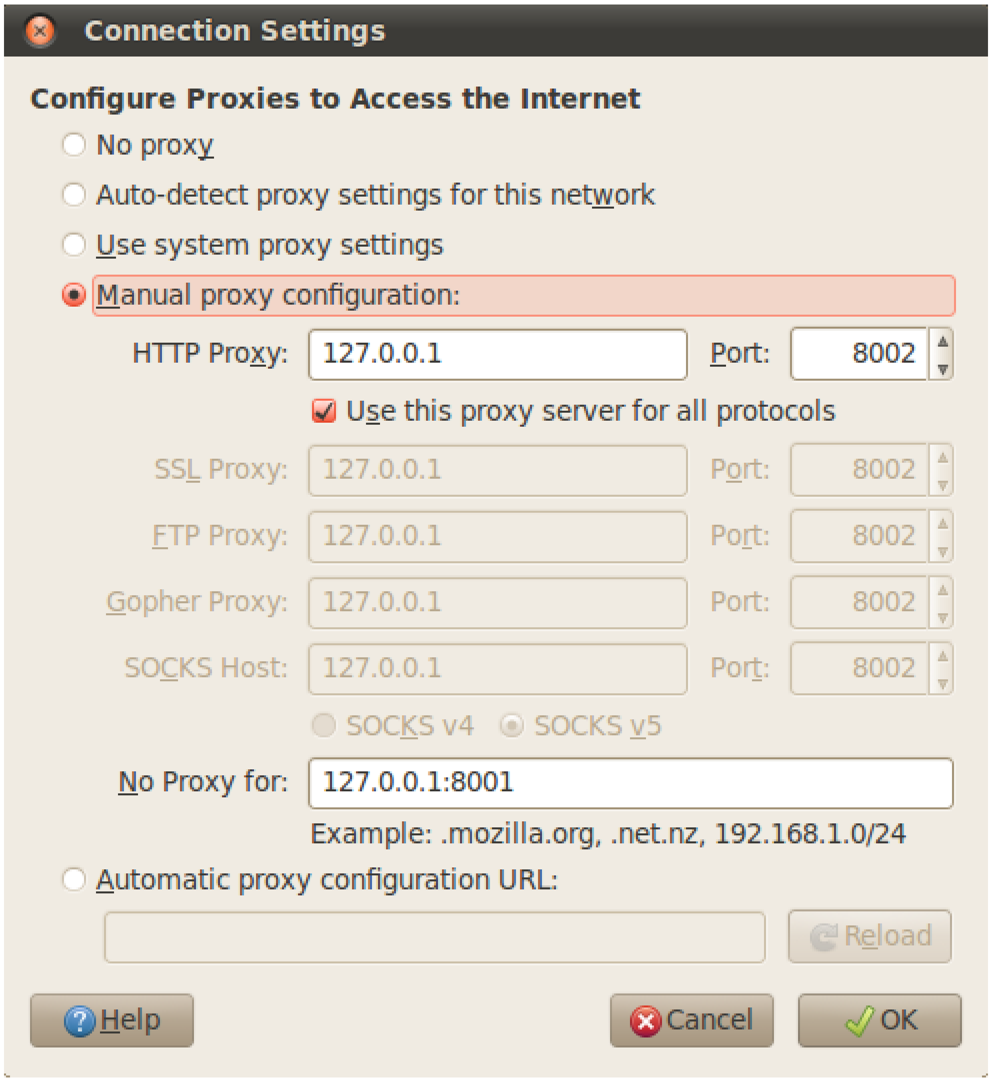
Start WebScarab ("Applications" menu, "Tools", "WebScarab").



Starting WebScarab

Set Firefox to proxy all traffic through WebScarab:

Edit (Menu) → Preferences → Network (Tab) → Settings.


Connection Settings

Configure Proxies to Access the Internet

No proxy
Auto-detect proxy settings for this network
Use system proxy settings
[✓ Manual proxy configuration:]

HTTP Proxy: 127.0.0.1 Port: 8002
[✓] Use this proxy server for all protocols
SSL Proxy: 127.0.0.1 Port: 8002
FTP Proxy: 127.0.0.1 Port: 8002
Gopher Proxy: 127.0.0.1 Port: 8002
SOCKS Host: 127.0.0.1 Port: 8002
SOCKS v4 [✓ SOCKS v5]
No Proxy for: 127.0.0.1:8001
Example: .mozilla.org, .net.nz, 192.168.1.0/24

Automatic proxy configuration URL:

Configuring Firefox to use WebScarab as a proxy

Select Manual proxy configuration, and configure the settings as above. Click OK.

Note that all of Firefox's Web traffic will now be sent via the WebScarab program, which is acting as a local proxy program. This enables you to view and modify any requests send to or received from the server.

Refresh the page in Firefox (F5), and view the information that WebScarab has collected.

Click the "Proxy" tab, "Manual Edit", and click the checkbox to "Intercept requests".

Now back in Firefox, refresh the page.

WebScarab will intercept the request and prompt you to alter the request before it is sent to the server. Experiment by making some changes. Click “Accept changes”.

Experiment with intercepting responses: uncheck the “Intercept requests”, and check the “Intercept responses” checkbox. Reload the page (Shift-F5), and note that you can view and modify the HTML source before it reaches Firefox. Make some changes to the HTML response and click “Accept changes”.

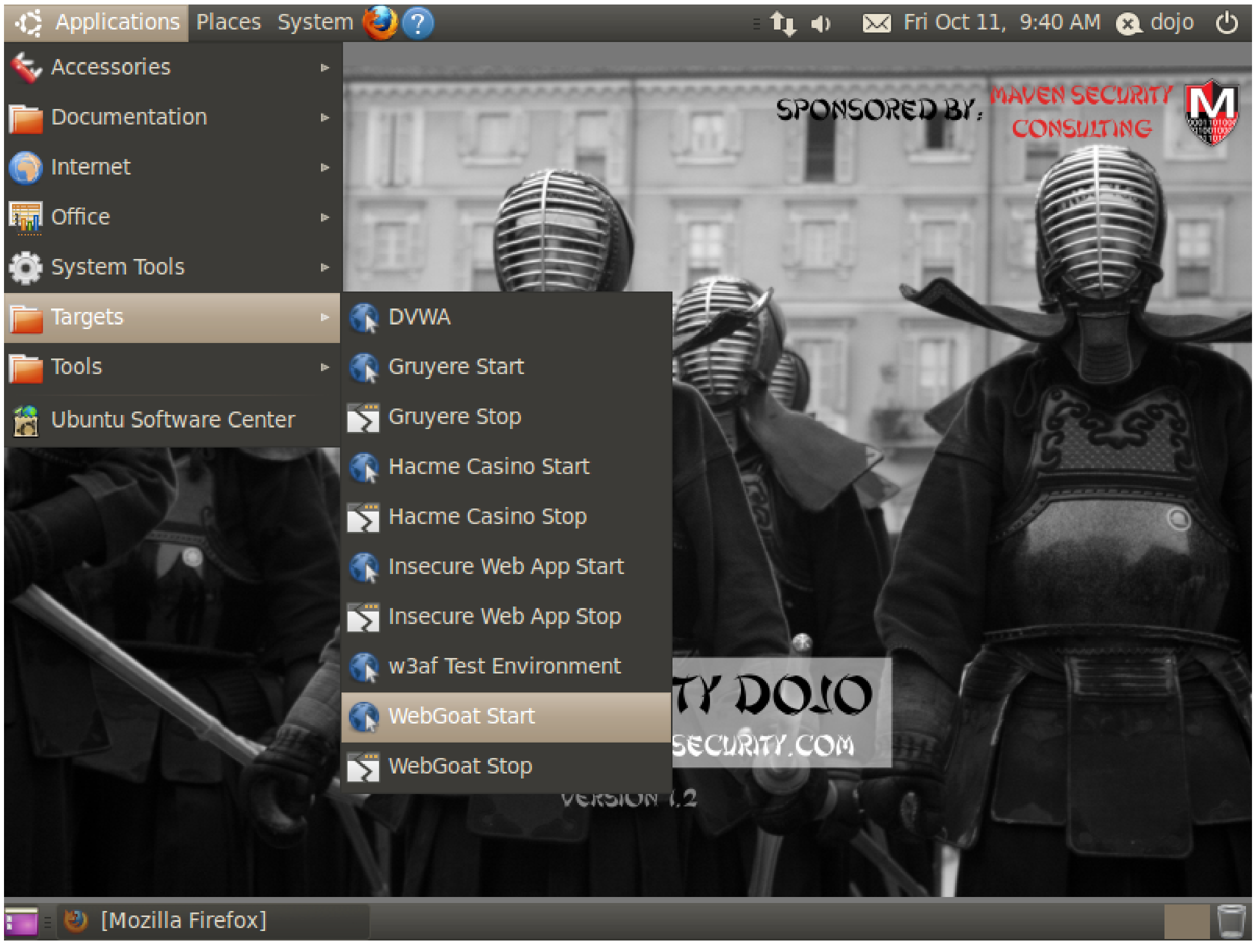
Turn off all intercepts, by unchecking the “Intercept requests” and “Intercept responses” checkboxes.

Using local Web proxies enables a security tester to feed unexpected data through to Web servers and analyse responses, potentially exploiting Web-based software vulnerabilities, conducting attacks such as SQL injection or cross-site scripting (XSS).

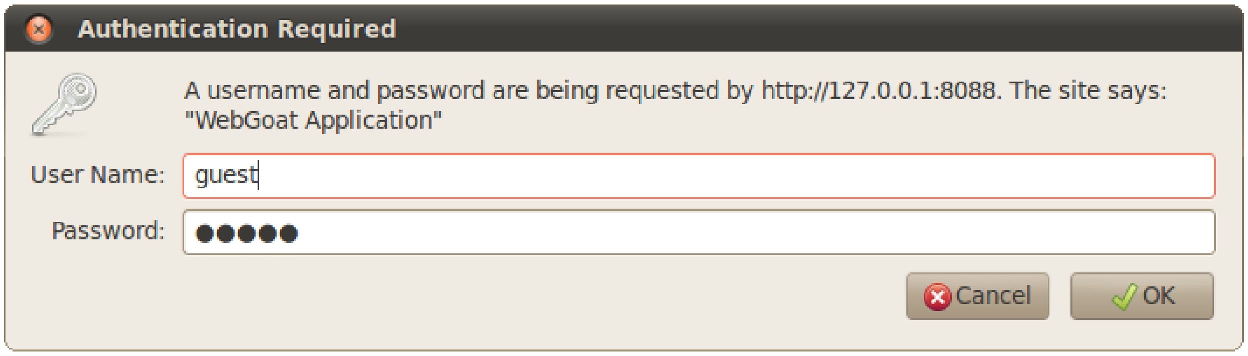
Working with WebGoat

WebGoat, developed by OWASP, provides a series of lessons on web security, by presenting the user with scenarios that include deliberately insecure code. You can attack each web vulnerability to pass each scenario.

Start Webgoat: Click the “Applications” menu, “Targets”, and “WebGoat Start”.



Starting WebGoat



Authentication Required

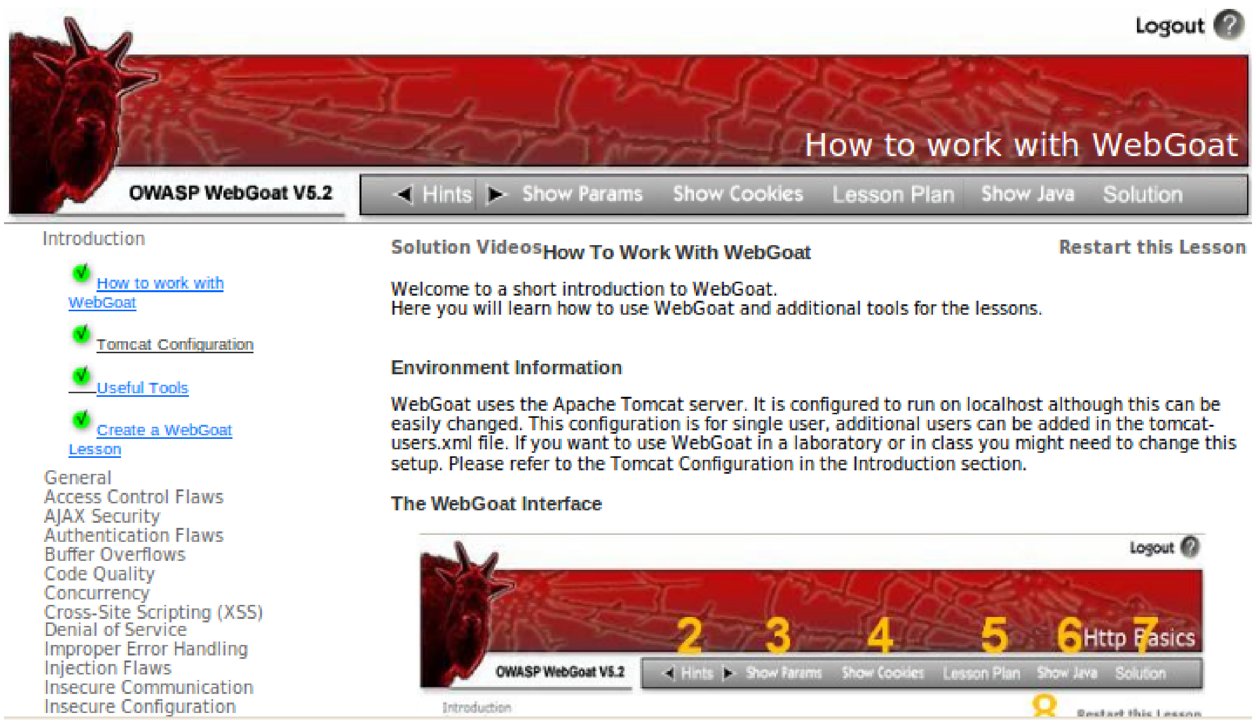
A username and password are being requested by http://127.0.0.1:8088. The site says: "WebGoat Application"

User Name: guest

Password: ●●●●●

Logging into WebGoat

Click OK, and then scroll down and click "Start WebGoat".



How To Work With WebGoat

Welcome to a short introduction to WebGoat.
Here you will learn how to use WebGoat and additional tools for the lessons.

Environment Information

WebGoat uses the Apache Tomcat server. It is configured to run on localhost although this can be easily changed. This configuration is for single user, additional users can be added in the tomcat-users.xml file. If you want to use WebGoat in a laboratory or in class you might need to change this setup. Please refer to the Tomcat Configuration in the Introduction section.

The WebGoat Interface

Working through WebGoat's scenarios

Read through the Introduction and Useful Tools sections.

Work through the "General" lab activities.

Work through the "Cross-Site Scripting (XSS)" lab activities.

Work through the "Injection Flaws" lab activities.

Conclusion

At this point you have:

- Learned Web fundamentals, and simulated the actions of a Web browser using Netcat
- Learned about important web security concepts
- Conducted stored and reflected Cross-Site Scripting (XSS) attacks
- Conducted SQL injection attacks

Congratulations!