EE6052 Web-based Application Design
CE4208/EE4023/ED5022 Distributed Systems

Exercise Sheet 4

Always log on as user "puser" with password "PUSER".

Objectives: Learn how to:

- use WebScarab to intercept and modify HTTP requests and responses.
- mount Cross-Site Scripting (XSS) attacks against vulnerable applications.
- use AntiSamy to avoid XSS attacks in a web-based application.
- use safe encoding to avoid XSS attacks.

Preparation:
Download the sample forum application from the module's web page and open the contained project in NetBeans.

Install WebScarab: Download the WebScarab installer file from the module's web page & double-click on the .jar file. Follow the installation guide - the default settings for WebScarab should be fine. Start WebScarab by double-clicking on the file "WebScarab.jar".

Configure your Internet browser for use with WebScarab: Go to the settings page and use a proxy with the settings host=localhost, port 8008. Make sure that the proxy **is not** bypassed for local addresses.

Ex4.1: *Inspecting HTTP Traffic with WebScarab*
Start the sample forum application in NetBeans - make sure that the project is opened in the browser for which you configured the proxy settings. Play with the application by using each feature at least once. Switch to WebScarab and investigate how all pages and request/responses are displayed in the summary page.

Ex4.2: *Intercepting & Modifying Request/Responses with WebScarab*
Switch to the "Intercept" tab in WebScarab and click the "Intercept Requests" tick-box. Perform requests in the sample forum application and investigate the content of the intercepted requests. Perform the following:

- Intercept a request that creates a new thread and use WebScarab to modify the title of the new thread.
- Intercept a request that adds a new message to a thread and modify the content of the submitted message.
- Intercept a request and modify the content of a cookie.
- Intercept any request and add a new cookie to the session.

Uncheck "Intercept Requests" and tick "Intercept Responses". Intercept a response to a "read a thread" request and modify the displayed messages.

Ex4.3: *Cross-Site Scripting*
Either start a new thread or add to an existing thread the following message:

```
Cross side scripting attacks include texts like
<script> alert("XSS attack!!!"); </script> in user
input that is displayed without being sanitised.
```

Display the new/modified thread in your browser - what is happening? If the attack does not work, try another browser (or modify the settings of your browser - please revert the settings afterwards again) - some newer browser try to deflect XSS attacks by default.

Ex4.4: *Installing AntiSamy*
Download the files AntiSamyJarFiles.zip and AntiSamyPolicies.zip from the module's web page. Create folder "C:\AntiSamy" and copy all the jar files from the AntiSamyJarFiles.zip archive to it. Also, copy all the jar files to the "lib" folder of Apache Tomcat ("C:\Program Files\Apache Software Foundation\Apache Tomcat 7.0.22\lib") or your

Glassfish Server ("C:\Program Files\glassfish-4.0\glassfish\lib"). Extract all XML files from the AntiSamyPolicies.zip archive and copy them to "C:\AntiSamy".

Within NetBeans, go to "Tools→Libraries" and select "New Library". Call the new Library "AntiSamy" and add all jar files from the folder "C:\AntiSamy" to it.

Download the archive "AntiSamySample.zip" from the module's web page and open the contained project in NetBeans. Select project properties and make sure, the AntiSamy library that you created is included in the project. Also, ensure that the line

```
Policy.getInstance("c:\\antisamy\\antisamy-slashdot-1.4.4.xml");
```

in the file "antiSamy.jsp" points to an existing file - adjust if necessary. Run the file and enter the text "`<script> alert("XSS attack!!!"); </script>`" in the entry fields. As you will see, the first time the XSS attack is successful, the second time the attack is unsuccessful. Analyse the code in the file "antiSamy.jsp" to understand how AntiSamy is used to sanitise user input.

Ex4.5: *Forum with AntiSamy*
Modify the sample forum application: use AntiSamy to secure the application against cross-side scripting attacks. Try to mount the XSS attack from exercise 4.3 (to simplify your task, add the comment to any of the messages created in index.jsp).

Ex4.6: *Secure Forum through re-coding*
Modify the solution to Exercise 4.5 by writing your own scanner: rather than removing script tags and other "dangerous" HTML components using AntiSamy, wirte code to replace them with "harmless" encoded versions (for example, replace "`<script>`" with "`&lt;script&gt;`"). Test your solution and demonstrate that no Cross-Site Scripting attack can be mounted.