Practical Sessions 5 and 6 – SQL Injection

WARNING: You must not use the techniques or tools from these practical sessions to test a web application without the permission of its owner. Doing so is likely to be illegal.

Introduction

Web applications provide services of great economic and social importance in areas including e-commerce, online banking and e-government. These applications and the sensitive data they collect, store and process are the target of frequent cyberattacks. As such, suitable development processes and testing techniques must be used to eliminate as many of the security flaws exploited by these attacks as possible.

In the last two PSEC practical sessions, you will use a range of techniques to test web applications for *SQL injection*, and you will learn how to prevent SQL injection. SQL injection is the most common type of injection attack, regularly identified as the highest security risk for web applications, e.g., see OWASP's[1] 2017 report on *The Ten Most Critical Web Application Security Risks*.

Preparation

To carry out the practical exercises, you will need a basic understanding of SQL and PHP. Gain this understanding or refresh your knowledge (e.g., using the http://www.w3schools.com tutorials for the two technologies) before the first of these practical sessions.

Also, read the following materials on SQL injection before the first practical, and use them during practicals:

1. The first seven pages (up to and including A1) of OWASP's 2017 report on *The Ten Most Critical Web Application Security Risks*. The report is available at https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf.

2. The section on 'Testing for SQL Injection' (pages 108-114) from OWASP's *Testing Guide 4.0*. The guide is available at https://www.owasp.org/images/1/19/OTGv4.pdf.

Tools

The practical exercises will use the following tools:

1. The OWASP WebGoat interactive environment for teaching web application security, https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project.
2. The OWASP ZAP integrated penetration testing tool for finding vulnerabilities in web applications, https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project.
3. A Ubuntu installation of the Linux-Apache-MySQL-PHP (LAMP) web application development and deployment bundle.

A PSEC VirtualBox virtual machine comprising these tools was set up for you to use during the practical sessions. Due to security considerations, connecting to the Internet is disabled on this virtual machine. **You must not attempt to change this setting.**

Take notes of your solution to each exercise, and of any findings and insights. You may need these for subsequent exercises and/or the assessment.

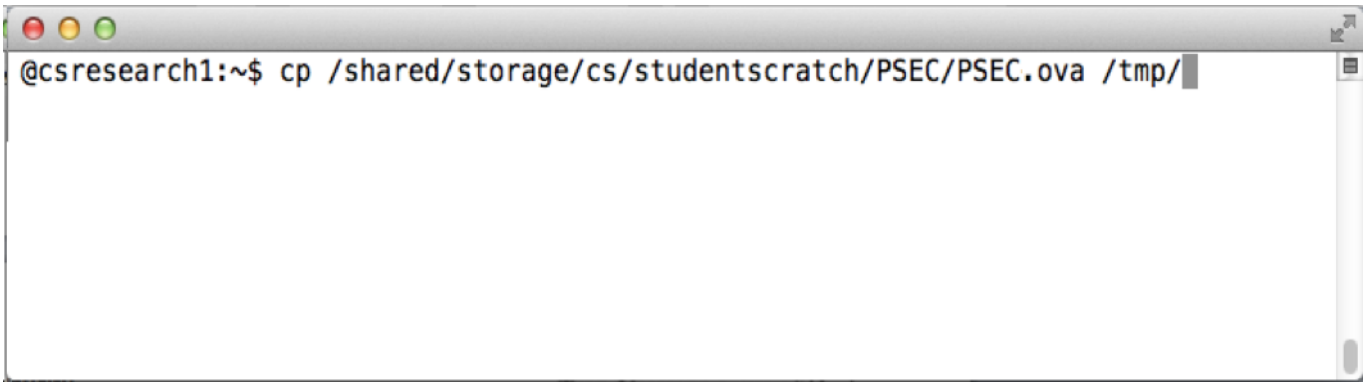
[1] The Open Web Application Security Project

Part I – Penetration testing

Exercise 1 – Setting up penetration testing

Start the PSEC virtual machine as follows:

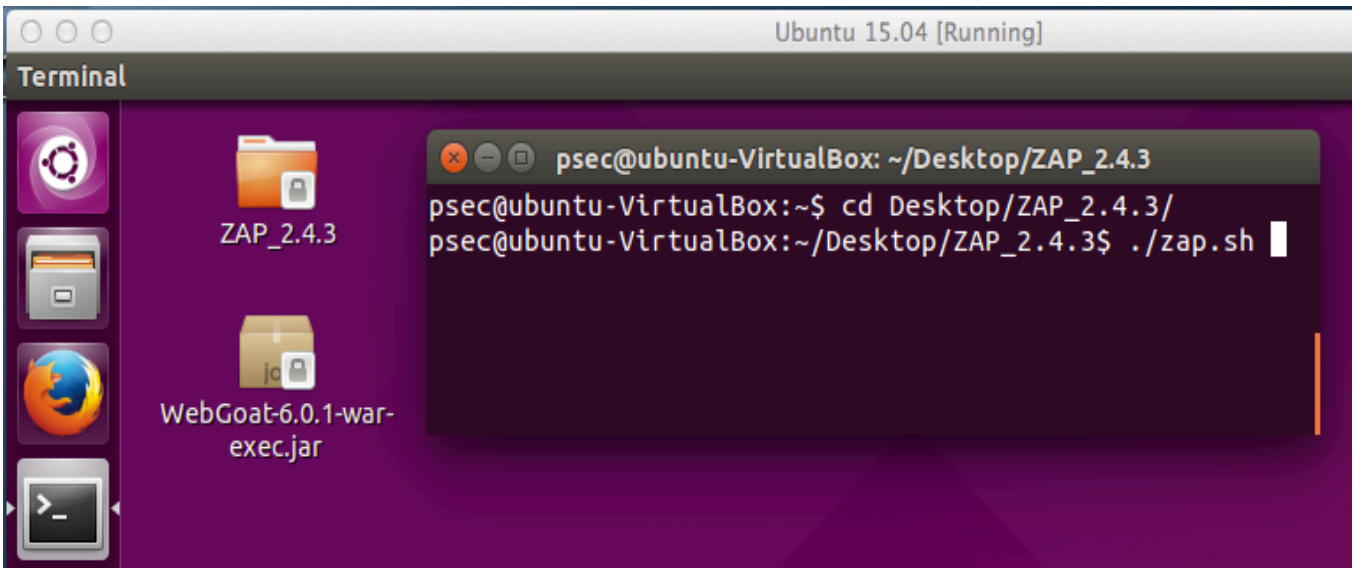
- Boot your lab PC into GNU/Linux. Note that this is not the default option, so you will need to select it shortly after you switch on the machine. If needed, reboot or restart the PC.
- Start a 'Terminal' window (from the list of applications installed on the PC – ask for help to locate it if needed), and run the command below to get a local copy of the virtual machine image used for the practical exercises. This is a 2.2 Gbytes file, so the operation will take up to 3 minutes – be patient.



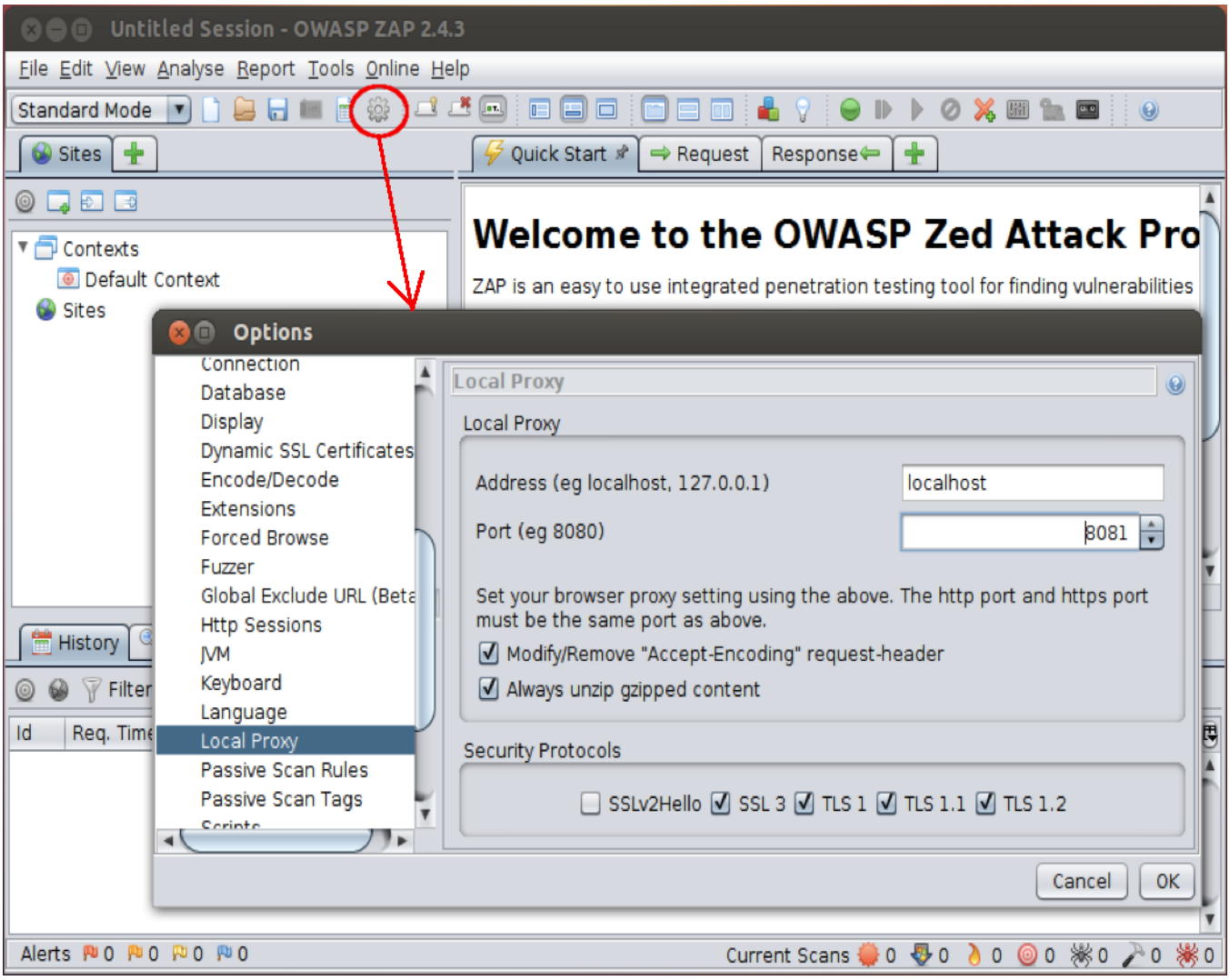
- Start VirtualBox (from the list of applications installed on the PC – ask for help to locate it if needed), and under File/Preferences/General, set the 'Default Machine Folder' to /tmp.
- Use the File/Import Appliance command from the VirtualBox menu to import the PSEC OVA file into VirtualBox.
- Start the virtual machine and log in as user 'psec' – the password for this account is TE$T:inject10n (note that sharing this password in the practical notes is acceptable, as the virtual machine contains no sensitive information).

Carry out the following setup operations on the virtual machine (see also the screenshots below).

1. Start ZAP in a Ubuntu terminal.

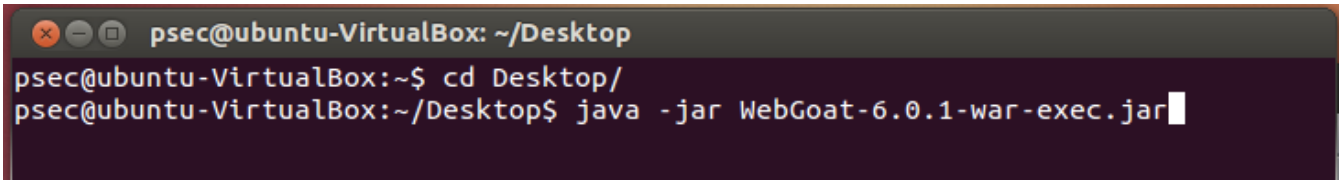


2. Under Tools/Options/Local Proxy, change the ZAP local proxy port to 8081 and click 'OK'.

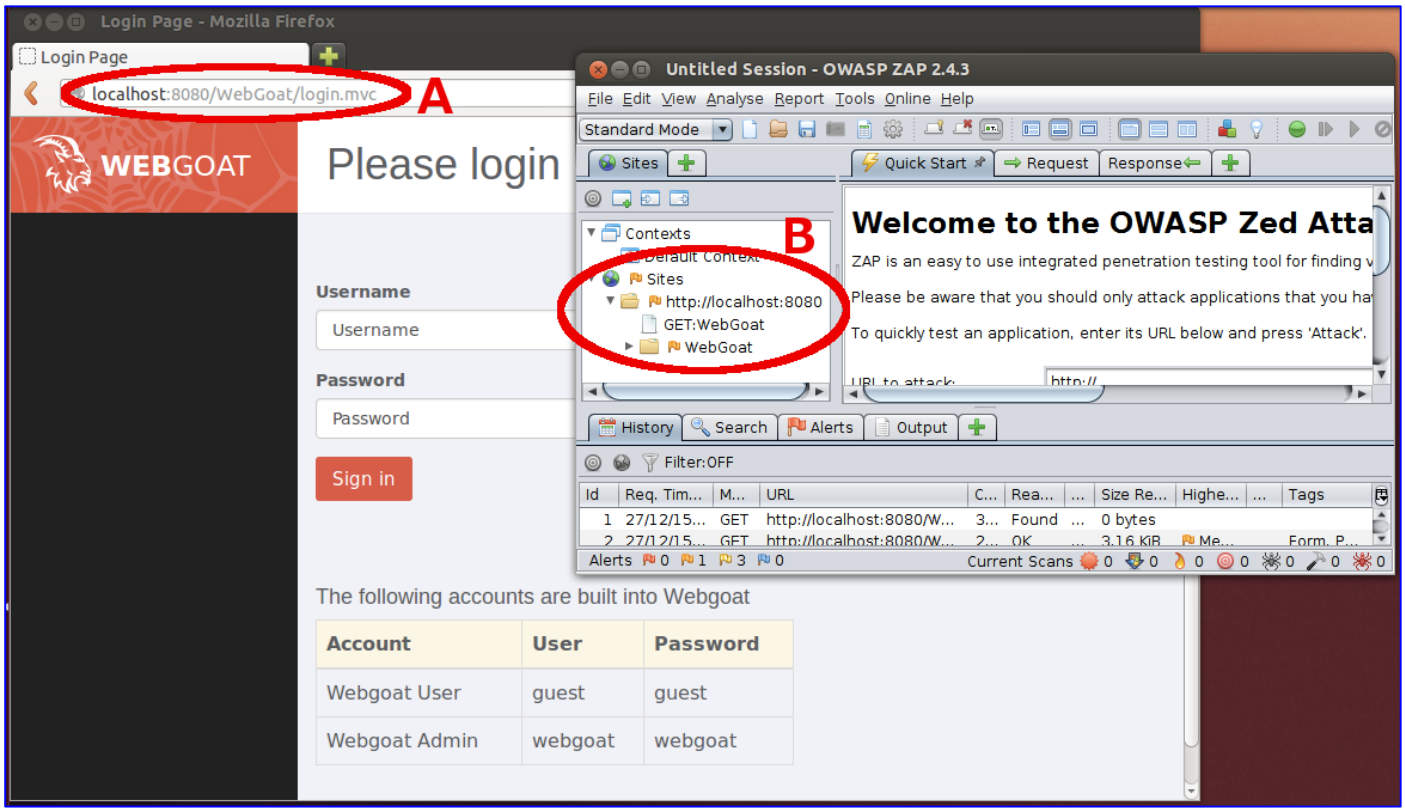


3. Start Firefox (click on the Firefox icon on the virtual machine desktop). Under Tools/Preferences/Advanced, select the Network tab and press the Settings button. Select 'Manual proxy configuration' and type in 'localhost' and '8081' as the HTTP Proxy and Port. Remove 'localhost, 127.0.0.1' from the 'No Proxy for' list, and click OK. This will set up ZAP as local proxy for Firefox (including for web applications running locally).

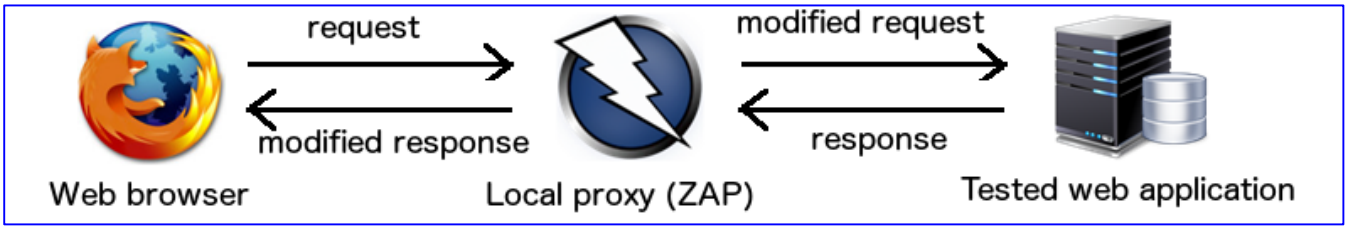
4. In a Terminal (click on the Terminal icon on the virtual machine desktop) start the WebGoat application by running the command below.



5. Load the WebGoat application at http://localhost:8080/WebGoat into the Firefox web browser (highlighted **A** in the diagram on the next page), and confirm that you set up ZAP as a local proxy correctly (highlighted **B**).



The setup for your penetration testing is now complete:



Which of the web application request and response do you expect to have to modify during the testing of a web application, and why?

Exercise 2 – Http and local proxy basics

Login on WebGoat as guest, briefly familiarise yourself with the application, and then select the General/Http Basics lesson from the menu on the left. After you type in a string in the 'Enter your Name' field and press 'Go!' to see the string reversed, set a ZAP breakpoint by right-clicking on POST:attack(…) in the 'Sites' ZAP pane.

Type another string into the input field and press 'Go!' to see the request in ZAP (you will notice the web browser "spinning" while waiting for the web application to respond). Modify the 'person' string in ZAP and press the 'Continue' button from the ZAP toolbar to forward the request to the web application.

Observe how ZAP also intercepts the web application response, and modify the label 'Enter your Name' in this response before pressing 'Continue' again to see the response reach the web browser.

Repeat these operations several times to familiarise yourself with the structure of the HTTP request and response and with the use of ZAP as a local proxy.

Exercise 3 – Numeric SQL injection 1

Remove or disable the ZAP breakpoint from the previous exercise.

Select the 'Injection Flaws/Numeric SQL injection' lesson from the WebGoat menu.

Use a suitable ZAP breakpoint to carry out the SQL injection described in the Lesson Plan displayed when you press the 'Lesson Plan' button at the top of the WebGoat web page.

To learn about web application penetration testing, make sure you give this a good try on your own, resisting the temptation to look up the lesson 'Hints' and 'Solution' from the beginning!

Remove or disable all ZAP breakpoints before moving on to the next exercise.

Exercise 4 – String SQL injection

Select the 'Injection Flaws/LAB: SQL Injection/Stage 1: String SQL Injection' lesson from the WebGoat menu.

Carry out the SQL injection described in the lesson plan, using OWASP's Testing Guide to design your "attack". To confirm that the SQL injection was successful, delete one of the user profiles (e.g., Tom Cat) and change Larry Stooge's credit card limit from 5000 to 8000.

You should only use the WebGoat hints and solution as a last resort (and to check your solution).

Exercise 5 – Numeric SQL injection 2

Carry out the SQL injection from the Injection Flaws/LAB: SQL Injection/Stage 3: Numeric SQL Injection WebGoat lesson. As before, try to come up with a solution without looking up the WebGoat hints and solution.

Exercise 6 – String SQL injection, data modification and addition

Without using ZAP breakpoints this time, carry out the "attacks" described in following 'Injection Flaws' WebGoat lessons:

- String SQL Injection
- Modify Data with SQL Injection
- Add Data with SQL Injection

Exercise 7 – Blind SQL injection

Read about the 'Boolean Exploitation Technique' in the section on 'Testing for SQL Injection' (pages 108-114) from OWASP's *Testing Guide 4.0*, and then carry out the two "blind" SQL injection attacks from the

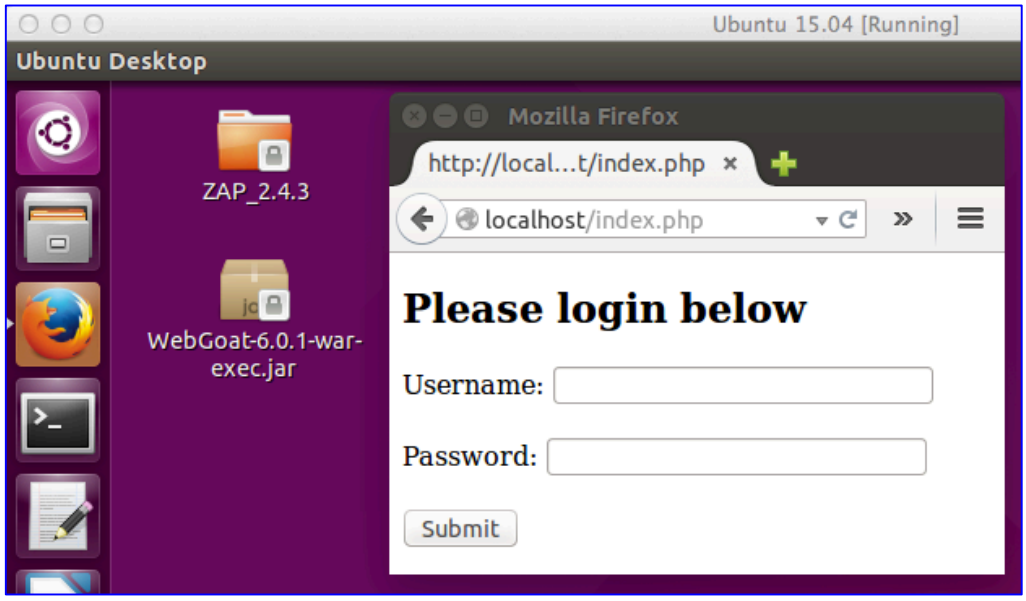
- Blind Numeric SQL Injection
- Blind String SQL Injection

WebGoat lessons. Note that you will need to use the 'Hints' for the two lessons because no 'Lesson Plan' is available. As before, resist the temptation to use the 'Solution' until you gave each problem a fair try!

Part II – Preventing SQL Injection

Exercise 8 – Identification of SQL injection vulnerability

Suppose you are a member of a software engineering team that developed a web application which uses the login web form deployed at http://localhost/index.php on the PSEC virtual machine (and shown below).



Use the penetration testing skills you acquired from the previous exercises to identify SQL injection vulnerabilities in this login form. In particular, is it possible to login as a registered user without knowing any username or password?

Exercise 9 – Fixing the SQL injection vulnerability

Read about the three SQL injection defence options in OWASP's "cheat sheet" on SQL injection prevention:

https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

Use Option #1 (*prepared statements*) to remove the SQL injection vulnerability from the login web form in Exercise 8. The PHP code for this web form is in the file /var/www/html/index.php, which you can edit on the virtual machine, e.g., using the PHP-enabled Ubuntu editor gedit. You should make a copy of this file before modifying it ('cp /var/www/html/index.php /var/www/html/index_backup.php').

To match the format of the other MySQL statements from index.php, **use MySQL prepared statements in the procedural style** described at

http://php.net/manual/en/mysqli.prepare.php.

A more general presentation of MySQL prepared statements in PHP is available at

http://php.net/manual/en/mysqli.quickstart.prepared-statements.php.

Make sure your code is correct by checking that you can log in as one of the registered users of the web application. You have full access to the MySQL database to examine the Users table (the MySQL root password is available in index.php), but one username you can try is bob with password MyPassword5.

Part III – Optional exercises

Exercise 10

Watch this short ZAP Tutorial on automated injection testing:

https://www.youtube.com/watch?feature=player_embedded&v=dqKGGCVFTvI

(please use headphones or keep the volume low).

Use ZAP's 'Active scan' tool as described in the tutorial to automatically identify vulnerabilities in the two web applications your worked with so far:

http://localhost:8080/WebGoat
http://localhost/index.php

Exercise 11

Automated testing is just one security control against injection threats. Other type of technique that are used to protect against injection threats are static analysis (of web application code) and run-time monitoring (of requests sent to web applications). Read about the AMNESIA tool that combines the two types of techniques in the research paper

William G. J. Halfond and Alessandro Orso. 2005. AMNESIA: analysis and monitoring for NEutralizing SQL-injection attacks. In Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering (ASE '05). ACM, New York, NY, USA, 174-183.
http://dl.acm.org/citation.cfm?doid=1101908.1101935

and on the AMNESIA website at http://www-bcf.usc.edu/~halfond/amnesia.html.

Exercise 12

If you have extra time, consider doing the following additional WebGoat exercises:

- Parameter Tampering/Exploit Hidden Fields
- Parameter Tampering/Exploit Unchecked Email
- Challenge/The CHALLENGE (Stages 1 and 2)

You may also use any spare time to learn more about security testing from OWASP's Testing Guide 4.0, available at https://www.owasp.org/images/1/19/OTGv4.pdf.