# 3 SQL injection

The following exercise will help you understand the SQL injection techniques and systematic vulnerability testing. While the main focus is on the SQL injection, there is also one Insecure Direct Object References vulnerability that you are supposed to exploit.

The first part of the lab exercise is based on the Hackxor webapp hacking game. The game consist of a set of sites that you are supposed to hack. The first two steps of this hacking game, which include *Wraithmail* (an e-mail service) and *Cloaknet* (an anonymizing proxy service), are used for this exercise.
If you want to try your skills directly, we encourage you to try solving these first two steps of the game without reading the lab instructions in the following section. You will, however, in any case have to answer all questions.

## 3.1 Preparation: Burp Proxy Setup

You will need an intercepting web proxy for the exercises, so before getting started, set up the burp proxy:

- Download the free version from http://portswigger.net/burp/proxy.html
- Unpack, `cd` to the folder with burp
- Run the proxy with: `java -jar burp...`
  (replace `burp...` with the correct filename)
- In Firefox select the Edit/Preferences menu, go to Advanced → Network → Settings:
  1. Tick "Manual proxy configuration:"
  2. Set
     - HTTP proxy: 127.0.0.1
     - Port: 8080
  3. Tick "Use this proxy for all protocols"
  4. Delete the "No Proxy for" line

In the Burb proxy software, go to the tab named `proxy`, and when the button named `intercept is on` is enabled (which it is by default) all requests to websites you visit in the browser will be intercepted and held by the proxy software. Once a request has been intercepted, you always have the possibility to change it (just edit the displayed text) before actually sending it out (by pressing the button `forward`).

## 3.2 Wraithmail

The game scenario is as follows: you are a hacker who got the task to track down a person who performed an attack on one of the Wraithmail accounts. For that, you should login to Wraithmail and read the message named "trace job".

**Wraithmail**

Website: `http://wraithmail.csc.kth.se:8080`
Username: `algo`
Password: `smurf`

In this first part, you will exploit an Insecure Direct Object References vulnerability. You are encouraged to try any other attack to check whether the site is vulnerable to any of them or not.

**Hints**

- Pay attention to the *referer* line in the "trace job" letter
- Explore the site, which functionalities (like composing a message, etc.) does it offer? Is one of them vulnearable? Note: You will need an intercepting proxy to exploit the vulnerability.
- If your attack succeeds, you will be able to see that the *abuse contact* is from *Cloaknet*.

**Answer the following questions:**

- Briefly describe the attempt attack of the hacker that you are tracking (no details, just the rough idea).

_____

- What username was used for the attack?

_____

- What was the logged IP address?

_____

**Milestone**

Report your progress to a lab assistant. _____

## 3.3 Cloaknet

By now, you have got evidence that the attacker used a proxy service called Cloaknet to hide its IP address. In order for you to find out who the attacker is, you should proceed with the Cloaknet website:

**Cloaknet**

Website: `http://cloaknet.csc.kth.se:8080`

There, your task is to perform an SQL injection attack to retrieve the account credentials (login, password, id,...) of all registered users at Cloaknet. You should also be able to confirm that the source of the attack was from `GGHB`, that the target was `wraithmail:8080/send.jsp`, and that the date matches the one you could see in the Wraithmail logs.

You might need to look at the SQL Injection Cheat Sheet mentioned in Section 1.1 to complete this task. Note that testing for the right way to get around the input filters can take some time. Try to test it systematically, but if you get stuck you can ask the lab assistants for hints.

**Guidance with questions:**

- Name at least 3 methods (in terms of the HTTP protocol) that are used to transfer data from the browser to the web application (on the webserver) and that can be utilized to perform injection attacks.

_____

- One of the values that goes as input to the SQL query in Cloaknet is *user* (the username that you type on the login page). Name two other values that might be used as input in the SQL queries. Hint 1: these two values are sent to the server with two different methods (see the previous question). Hint 2: Look at the two HTTP requests sent to the server when logging in.

- What do you think how many SQL queries are issued, when you login to the website? What inputs can these queries use? How do you think these SQL queries look like? Write them down as precise as possible (e. g., using SQL syntax/pseudo-code).

- Check whether the Cloaknet web application's inputs are vulnerable to SQL injection or not. How do you do that? Give two examples.

- Were you able to cause an SQL error or an HTTP error (500) to be displayed on the webpage? How did you do it? What information did you learn? What kind of database and web server are being used?

- Why is it useful to know which database is used?

- When you try to inject a meaningful SQL string, a first obstacle you might encounter is that white space characters are treated as delimiters in the cookie string in the HTTP header. How can you anyways use space characters in a cookie value?

- All inputs are filtered! The filter escapes some characters and removes some words (e. g., `SELECT`). Name at least four techniques to bypass filters in general (with examples). Hint: check the cheat-sheets mentioned in the introduction.

- Table and column names are very predictable in this lab. What could be the name of the table containing all usernames and passwords?

- The `UNION` operator is used to combine output from several `SELECT` statements. What requirements should these statements meet in order to be "unioned"? What do you have to do if the tables you want to union do not have the same number of columns?

- You should be able to infer the number of columns used in the query statement from the output you see on the page. Sometimes, though, not everything is shown, and then it is important to find out the number of columns. How can it be done? Provide an example.

- You can union tables even without specifying column names. How can you include all columns of a table in the `UNION` statement without explicitly naming them?

- What are the username and the password of the attacker? How did you get the list of usernames and passwords? How did you know which one was the attacker?

**Milestone**

Report your progress to a lab assistant.

## 3.4 Protection against SQL injection

There are many mitigation techniques for SQL injection.

### Questions:

- Describe how parametrized queries help protect against SQL injection. What other protection techniques can be used on the application's code level?

- What is a stored procedure? Describe how it protects against SQL injection. What other protection techniques can be used at the database level?

- What combination of protection techniques would you choose, to mitigate the SQL injection threat?

**Milestone**

Report your progress to a lab assistant.