

Introduction to IT Security

Homework 7 – SQL Injections

Abraham Murciano and Jacques Benzakein

September 30, 2020

1 RedTiger's Hackit

1.1 Level 1

The challenge for the first level is to obtain the password for the user Hornoxe. The webpage, whose interface is visible in figure 1, consists of two fields to enter a username and password, and some descriptive text on some category labelled '1'.

Welcome to level 1

Lets start with a simple injection.

Target: Get the login for the user Hornoxe
Hint: You really need one? omg -_-
Tablename: level1_users

Category: 1

This category does not exist!

Username:

Password:

Figure 1: UI of website for level 1

The first thing we tried was to enter some text into the username and password fields to see if our input was displayed anywhere in the site, or included in the page source. This path did not yield any success.

The next attempt was to input the text `Hornoxe' and sleep(20)` into the username field, and see if there was a twenty second delay in the processing of the submission. Unfortunately there wasn't, which meant that the username field was likely invulnerable to SQL injections. Trying something similar with the password field was equally unsuccessful.

The next thing to try was the category. After performing some inspection, we found that clicking on the link on the number 1 passed a url parameter to the page of the form `?cat=1`. When the page loads with that parameter, instead of saying "This category does not exist", it displayed some text about that category. This behaviour would lead us to believe that there is some SQL statement which retrieves the description of category 1 which looks something like this.

```
SELECT [some columns] FROM categories WHERE id=1
```

In order to confirm this suspicion, and to find out if this input was vulnerable to SQL injections, we requested the site with `?cat=1 and sleep(20)--` as the URL parameter. Luckily, the page took twenty seconds to load, so it seems that our suspicions are correct.

What we wanted to do next was to cause the where condition to fail, and then union whatever data we wanted (the password of user Hornoxe) to the query result, so the site would display that instead of the description of the category. However, in order to union something to the result, we needed to know how many columns were being selected, since the second query that we were going to union to the existing one needed to use the same number of columns.

So we tried to select one column, then two, then three, and when we tried with four we were able to see what we had selected on the page. The URL parameters looked like this.

```
?cat=0 union select 1,2,3,4--
```

After running this, we were able to see the numbers three and four displayed on the site, so we can infer that the third and fourth columns of the query result are what is displayed on the site.

Changing the query to the following one successfully displayed the password "thatwaseasy" on the page.

```
?cat=0 union select 1,2,username,password from level1_users--
```

1.2 Level 2

Here we are displayed with a simple login with a username and password field. Trying various malicious inputs in the username field was unsuccessful, which

led me to believe that the username field is not vulnerable to SQL injections. That left only the password field.

A typical SQL query to authenticate a user looks something like the following.

```
SELECT * FROM users WHERE username='[username]' and
password='[password]'
```

Assuming this was what the query in question looked like, we attempted to enter “' or 1=1--” as the password. However, that resulted in the page showing a PHP error which meant the query had failed.

We suspected that perhaps there was an important part of SQL code after the location which the password was being inserted, and since we had two dashes in the password, we were commenting that important code out. We therefore needed to modify our password so that we would not comment any code out, while still being a syntactically correct query.

The password “' or 1=1 or '” accomplished just that and successfully logged me in.

2 Zixem

2.1 Level 1

We are tasked with obtaining the version of the database engine and username of the database user. The page we are targeting has no input fields, but it does have a URL parameter `?id=1`. Replacing the URL parameter with `?id=1 and 1=2` causes some of the data on the page to disappear, which indicates that the query returned an empty result set.

Next, we tried to find out how many columns there were in the select statement that was being run. To do this we used the following URL parameters, each time adding more numbers to the end until we didn't receive an error. With this we learned that there have to be three columns selected in the union.

```
?id=1 and 1=2 union select 1,2,3--
```

This time, we were able to see the numbers one and two in the page content, so we replaced them with `user()` and `version()` like so.

```
?id=1 and 1=2 union select user(),version(),3--
```

This revealed that the version is 5.6.33-log and the user is zixem@localhost.

2.2 Level 2

In this challenge we are facing a website with a URL parameter of the form `?showprofile=4`. The site then shows the user ID, username, and age of the selected profile.

The first thing we tried was to replace the four with a single quote. That caused the site to display a MySQL syntax error message which occurred “near ‘’’ at line 1”. Since there were five single quotes in the error message, the first and last ones are part of the error message, used to quote actual SQL code. That leaves three quotes which were in the SQL query. This indicates that whatever input we give is put in single quotes, probably because the profile id field is of type string.

Next, we want the query to always fail, so that we can union our own data to the result set. The following URL parameter achieves that. The final single quote must be there to match the quote that was intended to go after the four.

```
?showprofile=4' and 0=1 and '
```

The next step is to determine how many columns are being selected. We requested the site with a few similar URL parameters, until one did not give me an error.

```
?showprofile=4' and 0=1 union select 'one','two','three','four
```

This showed ‘one’, ‘two’, and ‘three’ on the page, but did not show ‘four’. So finally we ran the following query to reveal the user and version.

```
?showprofile=4' and 0=1 union select user(),version(),'','
```

2.3 Level 3

This challenge is practically identical in structure to the previous one. Attempting to run a slightly altered version of the final URL parameter from the previous level gave the following error message.

```
...syntax to use near 'uni select user(),version(),'','' at...
```

As the error message shows, the word ‘union’ is being converted to ‘uni’. To combat this, we attempted to use the word ‘unionon’, and that worked. The final query is shown below.

```
?item=3' and 0=1 union select user(),version(),'','
```

2.4 Level 4

Again, this challenge was identical to the second one, other than there being a different number of columns. The first thing we tried was to use the same query as the one from Level 2, then when we were told there was a different number of columns, we added some more until we got the right answer. This is the final query.

```
?ebookid=7' and 0=1 union select user(),version(),'','',''
```