CS380 — Exercise 6

February 15, 2017

Due: Monday, February 20, 2017 before midnight (60 points)

You may work with a partner to complete this exercise. If you do work with a partner, only one person should create the codebank project and add the other as a member with at least developer privilege.

You must create a lab report and submit that report to `codebank.xyz` in a repository named `CS380-EX6`. You can make as many commits and push as many times as desired before the deadline.

Background

In this exercise, we will use a single virtual machine that will be running a web application that has SQL injection vulnerabilities. It is your job to exploit these vulnerabilities and document how you accomplish it.

For more information about SQL, you can read the tutorials available here: http://www.w3schools.com/sql/default.asp.

Setup

Launch one of the Ubuntu 12.04 SEED virtual machine images.

Note: the password for the `seed` user is `dees`. The root password is `seedubuntu`.

The first step will be to disable a built-in protection for SQL injection attacks in PHP. Edit the file `/etc/php5/apache2/php.ini`. Find the line `magic_quotes_gpc = On` and change the `On` to `Off`. You will need root privilege to edit the file. You can do this by launching the text editor with the `sudo` command as we did with the packet sniffer in the previous exercise.

Next, we will restart the apache web server that is running the php web application using:

```
sudo service apache2 restart
```

The web application should now be running. You can go to the website using the provided Firefox web browser by going to `www.sqllabcollabtive.com`. That hostname was directed back to the local machine using the `/etc/hosts` file so the URL will work only within the virtual machine. The web application is a project management website.

You should now be presented with a login prompt. Go ahead and login with username `admin` and password `admin`. On the web site, there is a project listed called *Users' Account Information* that lists all of the users and their passwords. Take note of this information and then logout using the button on the top right of the webpage.

SQL Injection in User Authentication

The web application's PHP code is located in `/var/www/SQL/Collabtive/` and specifically the file `include/class.user.php` has the code used for logging in a user.

Note: do not modify any of the web application's PHP code, you should be exploiting the vulnerability only through the forms in the web site.

The code for logging in looks something like the following:

```
$sel1 = mysql_query(
   "SELECT ID, name, locale, lastlogin, gender
    FROM USERS_TABLE
    WHERE (name = '$user' OR email = '$user') AND pass = '$pass'");

$chk = mysql_fetch_array($sel1);

if (found a record)
{
    allow user login;
}
```

The variables `$user` and `$pass` will be filled in from the login form.

This login code has an SQL injection vulnerability. Your first goal is to login as user `ted` without providing his password. Document your process for doing this and provide screenshots of the login screen with your input along with the web page after logging in.

Once you can login as `ted`, try the same process for the `admin` user. Can you login as the administrator without providing their password? Document the results and provide screenshots.

Finally, can you find a way to login with a blank username (i.e., the user name in the query is considered blank but the user field in the login form is not necessarily blank)? Document how you are able to do this. What happens when you login this way? Which user does the website think you are? Document the results and provide screenshots.

Bonus: SQL Injection in User Profile

You may complete this problem for up to 10 bonus points, but it is not mandatory.

In the web application a user can modify their profile by logging in then clicking *My Account*, then *Edit*. This page allows the user to fill in the information then some PHP code in `include/class.user.php` will put the information in to an SQL `UPDATE` statement. This `UPDATE` statement also has an SQL injection vulnerability. Look in the PHP code for how it constructs the `UPDATE` statement to determine how to perform the SQL injection.

Your job here is to log in as `ted` and use this page to change the password of `alice`. Document how you do this and provide screenshots. Show that you have logged in as `alice` with the new password.

Next, do the same but change the password of `admin` while logged in as `ted`. Document how you do this and show screenshots of logging in as `admin` with the new password.