**CyberSecurity Course Project I**

In order to make this more readable, the text of this essay is also included in the project on github as a word document. So, please first clone or download the project and you can continue reading a nicely formatted document which is PS\_essay.docx located at the root directory of the project.

Download instructions:

Reproduction instructions – general structure

* 1. Download from github or clone at <https://github.com/piet8stevens/csb-p> . Set the branch to master in the branch dropdown menu. This is the version of the software that includes the 5 OWASP issues.
  2. For each issue, follow the instructions given for that specific issue in order to reproduce it.
  3. Download from github or clone at <https://github.com/piet8stevens/csp-b> . Set the branch to fixed in the branch dropdown menu. This is the version of the software that has the 5 OWASP issues fixed.
  4. For each issue, follow the instructions given for that specific issue in order to demonstrate that the fixed branch indeed fixes the issue.

Finally, I conclude my essay with some observations and comments on how to do better next time.

1. A2+A4:+A7 Broken authentication and Session + Insecure Direct Object References (Leave a url parth unprotected - download an upload by changing the html.)
   1. Instructions to reproduce:
      1. Create new user "user1" with password pwd1.
      2. Login as user1 with password pwd1.
      3. Upload file 1 "test1.txt"
      4. Logout
      5. Login as user roger password carrots
      6. Upload file 2 "test2.txt"
      7. In the html, modify /files/2 to /files/1. Click on test2.txt. Test1.txt is downloaded. You have access to user1's file.
2. A3: Cross-site scripting (XSS)
   1. Added message functionality to the app.
   2. Having a hard time to get javascript to execute. A la webgoat. Somehow, wasted a full day trying to do it in JS or to use JS to execute the message. Finally, when I rewrote the html, it just worked. No, it worked because of executeJS. Figured out how to get <script></script> correctly added but could not get it to execute automatically like webgoat. Developed 3 kludges and will include the simplest one in my "solution".
   3. Instructions to reproduce
      1. Create new user "user1" with password pwd1.
      2. Login as user1 with password pwd1.
      3. Create a new message with title "IMPORTANT MESSAGE - READ URGENTLY" and content "<script>alert(document.cookie);</script>".
      4. Log out.
      5. Log in as user "roger" with password "carrots".
      6. Click on the message with title "IMPORTANT MESSAGE - READ URGENTLY". The popup shows that javascript was excuted, proof of stored xss attack being successful.
3. A5: security misconfiguration
   1. Instructions to reproduce:
      1. Start the application
      2. Use the following url in your browser: localhost:8080/h2-console
      3. Accept the default user and password, which are already filled in. You now have access to the entire database underlying the application because the h2 settings are insecure.
4. A8: CSRF- by loading csrf.html - can download from the side. Still to do : delete from the side.
   1. While logged in as roger carrots with test2.txt uploaded as /files/2, open another tab in the same browser.
   2. Edit file csrf-del-a.html and ensure that the /files/<nn> in 2 locations in the file has the correct number for <nn>.
   3. Open file csrf-del-a.html (typically use ctrl-o and select that file which is in the github clone top directory).
   4. Notice how the file is downloaded and the upload is deleted. This shows CSRF success.
5. A9: Using Components with Known Vulnerabilities: use outdated library
   1. Go to the root directory of the cyber-security-project
   2. Ensure you have maven installed and on your PATH. If not, find your system-specific installation instructions and install so you can access in the terminal (or for windows: from the command window).
   3. Start a terminal there and type "mvn dependency-check:check"
   4. You will receive a warning that "One or more dependencies were identified with known vulnerabilities in cybersecuritybase-project:" with details on which vulnerabilities ( a list of CVE's in the form CVE-<year>-<number>).