To make this readable, this essay is also included in the project on github as a word document. So, please first download the project as per the instructions and you can continue reading a nicely formatted document: PS\_essay.docx, located at the root directory of branch fixed.

I have selected OWASP vulnerabilities from the 2013 list, not the 2017 one.

**Download and overall reproduction instructions:**

* 1. Download from github or clone at <https://github.com/piet8stevens/csb-p> . Set the branch to “master” in the branch dropdown menu before you download or clone. This is the version of the software that includes the 5 OWASP issues (actually more than those 5 only).
  2. Now set the branch to “fixed” in the branch dropdown menu. This is the version of the software that has the 5 OWASP issues fixed.
  3. Both the faulty version and the fixed version can be run via NetBeans exactly like the TMC code. So, after downloads (and unzip if necessary), please start up NetBeans, and use File/Open Project to open the downloaded projects. The faulty version will be called cybersecuritybase-project .The fixed version will be called csb-p-fixed.
  4. For each issue, follow the instructions given for that specific issue in order to reproduce it.
  5. For each issue, follow the instructions given for that specific issue in order to demonstrate that the fixed branch indeed fixes the issue.

**Reproduce OWASP vulnerabilities and check that they have been fixed:**

First start up the faulty project and the fixed project:

1. Startup (see b.) both cybersecuritybase-project (faulty) and csb-p-fixed (fixed) in Netbeans. The project directory for the fixed project is referred to as <root-fixed> in the remainder of this text.
2. In order to start the server for a project:
   1. Select “run” (build will automatically happen the first time).
   2. If a popup menu “select Main Class for Excecution” shows up, click the button “Select Main Class”.
   3. Make sure you have 2 runs active (lower right-hand corner of NetBeans, you should see “1 more”).
3. Open a tab in your browser and go to localhost:8080 for faulty.
4. Open another tab in your browser and go to localhost:8081 for fixed. Most loaded pages in the fixed version of the project have “- fixed” added to their page title so as a tester, you can easily see whether you are in the right server (in addition to the port number of course).
5. Now follow the instructions for each vulnerability.

**Detailed instructions per vulnerability:**

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. Reproduction:
      1. Select tab localhost:8080.
      2. Type name “user1”in the Name: field and any address in the address field and click “Submit”.
      3. Create new user "user1" with password pwd1 by entering the password in the Password: field and clocking “Create new user”.
      4. Login as user1 with password pwd1.
      5. Upload file 1 "test1.txt" by clicking on “Choose File”, going to <root-fixed>/data, selecting “test1.txt” and then clicking “Add!”
      6. Logout
      7. Login as user roger password carrots
      8. Upload file 2 "test2.txt"
      9. Start developer tools in your browser (in chrome under “Menu/More tools/Developer tools”)
      10. Right click on the link “test2.txt”, select “Inspect”. Right click “/files/2” in the tools pane; using “edit attribute”, modify it to “files/1”.
      11. Click on test2.txt on the web-page.
      12. Test1.txt is downloaded. You have access to user1's files.
   2. Fix-check:
      1. Select tab localhost:8081.
      2. Execute instructions i) through xi) above.
      3. Now, a blank page shows and nothing is downloaded. The vulnerability is no longer there.
2. A3: Cross-site scripting (XSS)
   1. Reproduction
      1. Select tab localhost:8080/top.
      2. Create new user "user1" with password pwd1. Note: if you do not restart the servers, user1 still exists from the previous section, so you can skip this step.
      3. Login as user1 with password pwd1.
      4. Create a new message with title "IMPORTANT MESSAGE - READ URGENTLY" and content "<script>alert(“Hello! Nasty surprise!!”);</script>".
      5. Log out.
      6. Log in as user "roger" with password "carrots".
      7. Click on the message with title "IMPORTANT MESSAGE - READ URGENTLY". The popup shows that javascript was executed, proof of the stored xss attack being successful.
   2. Fix-check:
      1. Select tab localhost:8081.
      2. Execute instructions i) through iv) above.
      3. The pop-up does not show. No stored xss attack executed.
3. A5: security misconfiguration
   1. Reproduction:
      1. Use the following url in your browser: localhost:8080/h2-console
      2. 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.
   2. Fix-check:
      1. Select tab localhost:8081/h2-console.
      2. Accept the default user and password, which are already filled in. You now do not have access to the entire database, proof of the security misconfiguration not being there.
4. A8: CSRF- by loading csrf.html - can download from the side.
   1. Reproduction:
      1. Select tab localhost:8080.
      2. Log in as roger carrots and upload test2.txt if it is not there yet.
      3. Verify under which file number test2.txt has been uploaded by right-clicking on the test2.txt link and selecting “inspect”. The Elements tab will show you href=”/files/<nr>”, with <nr> having a value like “1” or “2”etc… . Use that <nr> in the next step
      4. Edit file <root-fixed>/data/csrf-del-a.html and ensure that the /files/<nn> in 2 locations in the file has the correct number for <nn>, which is <nr> from the previous step.
      5. Open file csrf-del-a.html (typically use ctrl-o and select that file which is in the github clone top directory).
      6. Notice how the uploaded file test2.txt is deleted and user roger has no more such uploaded file. This shows CSRF success, ie., the vulnerability is there.
   2. Fix-check:
      1. Select tab localhost:8081.
      2. Execute steps ii) through v) as above, but now use <root-fixed>/data/csrf-del-a-fixed.html
      3. Note how the CSRF token is absent and therefore the attack failed.
5. A9: Using Components with Known Vulnerabilities: use outdated library
   1. Reproduction:
      1. Open a terminal and 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>). Notice for example how in the faulty project (with older dependencies), the Jackson-core has 2 CVEs.
   2. Fix-check:
      1. Select tab localhost:8081.
      2. Execute mvn dependency-check:check in <root-fixed>.
      3. Notice how Jackson-core now has 1 CVE. The CVE-2016-3720 has disappeared and apparently fixed. Notice also that for some other dependencies, actually the number of CVEs has increased or that new dependencies show up with CVEs.

Thank you for any detailed feedback as there are quite a few kludges in this and I have struggled quite a bit with the spring boot framework as opposed to focusing on software security.