Name: ______________________________

Lab 6: Summary Lab

Exercise Objectives: The primary objective of this exercise to test students' ability to recognize and exploit a range of vulnerabilities in web applications.

Directions:

Start by downloading the code found at https://github.com/profh/67327_Arbeit and getting it set up on your machine. Be sure to read the README file that comes with the project (appears right on the first page of the github repo at the link above) as it has some information about setting up the application.

Finding vulnerabilities. There are at least a dozen vulnerabilities or security issues with this application. Identify each of these issues, exploit them if you are able (<u>and take a screenshot of the successful exploit whenever possible</u>) and record your findings on the template provided in the `doc/` directory of the project. If you wrote an external script to assist with the exploit, be sure to include that with the findings.

Submitting your findings. At some point after October 13th but prior to 11:59am on October 17th, all students must submit their findings to a drop spot at http://pawn.hss.cmu.edu/~67327/submit_exploit.php. All supporting materials must be archived as a .zip file and that file must be given the name `<YOUR_ANDREW_ID>_lab6.zip` – failure to follow this naming convention is an automatic 25-point penalty. There is a maximum file size of 16MB, so reduce photo resolution if you after zipping the file you are still pushing past that size limit. *No late submissions will be accepted without written excuse of medical emergency or other extenuating circumstances.*

Other notes of interest regarding this lab:

1. As stated in the course syllabus, this lab is to be done by individuals. There is to be **no collaboration between students** and any collaboration will result in a failing grade for all parties involved.

2. Additional rows can be added to the findings template if needed.

3. Do not look at the Ruby source code (but by all means look at the HTML and javascript code for each page) in doing your evaluation. If you record a vulnerability, you must be able to show apart from the Ruby code how it was discovered.

4. There are many vulnerabilities in the application – some large and some small – record as many as you can. Note that there are multiple versions of some vulnerabilities (e.g., just because you find one type of SQL injection doesn't mean that other types are not available). Likewise, one vulnerability might make other vulnerabilities possible.