First Class Project

First handin due: 11:59 pm, Saturday March 14th
Second handin due: 11:59 pm, Friday March 20th

The Project

You have to find security holes in our implementation of a course management app. The app is presented to you as a blackbox - you are given the usernames and passwords of regular unprivileged users to see how the site works. Discover how it’s supposed to work, then figure out how to break in. Regular users can quickly become a problem if they can escalate their privileges to that of an administrator.

Running the Web App

Each student gets their own instance of the web site to attack. In order to access the website, run ssh-firstclass [<local-port>]. This will open a port (8080 by default) on the local machine which you can use to access the website (ie, by visiting localhost:8080) in a browser. If you want to access the website from home, you can ssh into the cs department using the -L <port>:localhost:<hostport> flag, where <port> is the port to connect to on your local machine, and <hostport> is the port you specified as an argument to ssh-firstclass (or 8080 if you didn’t give an argument). Note that this will only work if you are also using the VPN. Alternatively, if you are not using the VPN, you can X-forward (using the -X flag to ssh) and run a browser from the command line (ie, iceweasel, chromium, etc).

First Handin: Exploits/Remediation

You should discover 5 distinct exploits. Each exploit must allow you to perform an action in the web application that you are not supposed to be able to do. For example, changing a grade, logging in as an admin user, deleting a particular user’s data, or accessing another student’s grades would all count as a successful exploit. Social engineering, phishing, and denial of service do not count as exploits. As a bit of clarification, we are looking for distinct classes of exploits. The primary rule for determining if an exploit is distinct from another is whether they require separate fixes. The same exploit on two pages of the application are not distinct exploits. You may use knowledge gained by any one exploit to inform any other exploits, but say for example that one exploit let you see everybody’s password, then logging in as that user does not count as a new exploit. Each exploit must function independently. If you have any doubts, please ask the TAs in case we may not agree about the validity of your exploits. All exploits must be implemented with source code handed in, or detailed instructions on how to execute the exploit.

For each exploit you discover, you should explain (without implementation) how to defend against this exploit. Try to come up with a fix that would have a low technical and economic cost if possible. For example, hiring experienced web app consultants to do it for you would not be a good fix for a simple attack. You may also argue that your exploit has no viable fix. Be sure to justify such an assertion.

Second Handin: Fixes

Once the first handin deadline has passed, you will be given the code for the application you attacked. For each of the 5 exploits that you have discovered, you should make changes to your application so that it is
no longer vulnerable to attack. Your fixes should be complete; it should not be possible for an attacker to bypass your patch and exploit the same vulnerability. If you described more than 5 exploits in your first handin, feel free to pick any 5 to patch. If you identify new vulnerabilities based on your code analysis, you may choose to document and patch them as part of your handin. You must document your changes in a README for this handin, or you will not receive any credit for your changes.

Grading/Extra Credit

This assignment will be graded out of 100 points, with each handin contributing 50 points to the total. If you submit an extra working exploit (that has a good description of the fix) before the first handin, you will be given 3 extra credit points. If you submit an extra patch (that is documented in the README) before the second handin, you will be given 2 extra credit points. The maximum score you can receive is 115/100.

Note that since the code for the website will be released after the first handin deadline, THE FIRST HANDIN DEADLINE IS A HARD DEADLINE. We will not accept ANY late submissions. You may not use late days on the first handin.

Hints, Tips, and Tricks

Accounts

Each student of the course has a login for FirstClass. Your username is your cs username, and your password is \texttt{iam<login>}, i.e. login: \texttt{jes pwd: iamjes}. Remember each of you have your own copy of the app, so feel free to log in as each other or yourself. Each member of the staff also has a login with their cs login, but the passwords are not known to you. They are, however, poorly chosen passwords. They are not simply random short alphanumeric passwords. They are more realistic, but certainly not passwords any of us would use in real life (especially after taking this course!).

Assumptions

Here are some hints:

- Assume the site is active. Users often log in, leave comments, visit profiles, and generally use all of the features of the site.
- When thinking about how you would exploit a vulnerability, imagine you can get any particular person to click on a link via some kind of trickery. Basically, you can send an email to any other student or staff member and we will click on it.
- Most parts of the application have been (intentionally) implemented poorly. That is, many vulnerabilities exist.
- All grades, comments, and handins are randomly generated, so don’t feel bad if your login has bad grades on it. Obviously this reflects nothing about you.

Where to Start

It’s vital that you first thoroughly see what the features of the application are. Figure out what you can do and as much about how it works as you can. Click on every link, check every box, and visit every page. If you are having problems thinking of more exploits, first look at lectures for ideas. You may want to get familiar with basic HTML and investigating the source code of particularly interesting pages. Get to know web developer tools like Firebug\footnote{http://getfirebug.com/} and Inspect Element. Think about what power you as a user have to
modify the application beyond the ways that the developers expected you to. It may help, as with many puzzles, to take a break or go to sleep to let your idle mind think more creatively about the problem.

**PHP-specific security knowledge**

For the vast majority of you, this project will be the first time you are analyzing and trying to fix an insecure PHP application. If you find a function and don’t know what it does, feel free to look it up in the PHP documentation (php.net). To assist you in getting started, here are some useful PHP functions/classes related to security:


**Tools and Rules**

**Tools**

Ask the TAs before moving forward with any broad external tools. Authoring tools such as Eclipse are of course acceptable, as are applications that proxy web traffic and allow you to inspect requests/responses and modify/replay requests (like Burp Suite as specified below). More fully-featured web application security analysis tools, however, should not be used since the purpose of the project is for you to discover vulnerabilities by yourself with your knowledge and skill as CS 1951-E students. Specific tools such as Firebug or Chrome’s “Inspect Element” tool in Chrome and Firefox are perfectly fine. If you have any questions about other tools, please ask TAs before using them.

**Burp Suite**

Burp Suite is a program used to analyze the security of web applications. The feature of the program which will be useful for this project is the ability to view and intercept packets sent over HTTP. If a packet gets intercepted, Burp Suite allows you to edit it before it gets sent off to the server. This feature is called Burp Proxy.

To launch Burp Suite, you can run `/course/cs1951e/bin/burpsuite`, or just `burpsuite` if `/course/cs1951e/bin` is in your PATH. Burp Suite is written in Java. If you have any issues running our script (which uses `/usr/bin/java`), you can run the .jar file directly: `java -jar /course/cs1951e/bin/burpsuite_free_v1.6.jar`.

Here is the link to documentation for Burp Proxy: [https://support.portswigger.net/customer/portal/topics/720233-burp-proxy/articles](https://support.portswigger.net/customer/portal/topics/720233-burp-proxy/articles)

You should first go through “Getting Started with Burp Proxy”, and complete the prerequisites. In particular, be sure to configure your browser to work with the program.

**Collaboration**

Large parts of this assignment can be completed easily once an idea is known. Therefore it is once again important to NOT discuss anything about this assignment with other students. Please do not discuss particular tools, particular parts of the app that you have attacked, or generally anything about your progress. It’s simpler to say nothing than to try and tiptoe around what is OK and what would be unfair help, or even damage, to other student’s work. Direct all questions to TAs via email or at hours.
Handin

First Handin
You should put together a thorough description of your exploits along with any code you used and a description of how to perform the attack. Document your exploits in a README and hand it in using \texttt{cs1951e_handin firstclass1}. A TA will evaluate your handin and tell you if the issues you’ve identified are sufficiently distinct and well-documented.

Second Handin
You should be turning in a patched copy of the code provided to you. In addition, please provide a README file which documents the following:

- If you identified and patched any vulnerabilities which you did not discover as part of the first handin, please document them here (include a thorough description of the vulnerability along with any code you used and a description of how to perform the attack).

- For each vulnerability, what changes you made to the code.

You should hand in your completed work with \texttt{cs1951e_handin firstclass2}