Electronic Commerce Virtual Laboratory

Joel Coffman  
Department of Computer Science  
University of Virginia  
Charlottesville, VA 22904  
jcoffman@cs.virginia.edu

Alfred C. Weaver  
Department of Computer Science  
University of Virginia  
Charlottesville, VA 22904  
weaver@virginia.edu

ABSTRACT

Website security is essential for successful e-commerce ventures, but the vital “how-to” components of security are often lacking in academic courses. This paper describes our attempt to instill an awareness of security concerns and techniques by having the students develop an Artist eXchange website, a social networking site that permits the posting and sharing of pictures, music, and text, including an end-user rating system. The six-homework set progresses through HTML, JavaScript, PHP, MySQL, file uploads, and security testing. An innovative feature is that each assignment is evaluated via automated testing, which guides the student toward detecting and correcting mistakes, especially with regard to common attack vectors.

Categories and Subject Descriptors
K.3.2 [Computers and Education]: Computer and Information Science Education—Computer science education, Curriculum

General Terms
Design, Security

Keywords
Electronic commerce, software laboratory, virtual labs, hands-on programming, guided instruction, evaluation

1. INTRODUCTION

Successful e-commerce depends upon many elements, not the least of which is well-trained computer scientists to design, maintain, repair, and innovate e-commerce technology. Over the past four years, we developed materials to support the education of computer science students who are prepared to work on the components of e-commerce stores. What we learned from that project—to be discussed later—points in the direction of a significant need for expanded education focused on the security of e-commerce systems.

The need for a focus on security in e-commerce education is consistent with a broader trend in computing. Security has been identified as a grand challenge in the field. At the CRA “Conference on Grand Research Challenges in Computer Science and Engineering” (23–26 June 2002), one of the five grand challenges selected was “Systems You Can Count On.” Quoting from the conference presentation on that topic [1]:

Today, information technology is the weakest link in many of our most critical applications and services. Digital computing and communications increasingly pervade our lives, our economy, and our nation’s critical infrastructure. In many applications this technology simply can’t be trusted; it creates problems that range from chronic aggravation to unacceptable vulnerability.

In his keynote address at ACM SIGCSE in March 2004 entitled “What is Information Security?” [5], Professor Eugene Spafford from Purdue University identified information security as one of society’s grand challenge problems for the next decade. Along with security, a pervading theme in these discussions has been the need to educate our students about their ethical responsibilities to themselves, their employers, and society at large. There is simply no doubt that security is a crucial topic that must be taught as part of the computing discipline in general and in the context of e-commerce in particular.

1.1 Existing Course Materials

In keeping with the work of Weaver [9, 8, 10], we adopted a software laboratory to support a computer science course on e-commerce technologies. Core material is first presented in lecture and then immediately reinforced with lab exercises. In an initial course offering, all students found physical labs helpful, but the overwhelming majority (87%) disliked the fixed lab times, preferring instead to do the work on their own schedule. These virtual labs, which may be completed at the student’s own pace, center around ten web technologies—including HTML, JavaScript, PHP, ASP, and Flash—that students pursue based on their personal experience. Students who feel comfortable with the material may skip immediately to the intermediate or advanced exercises on each topic. Due to the virtual labs being viewed as a resource rather than a requirement, none of these exercises are graded; instead, a series of homework assignments and a course project are responsible for integrating the disparate concepts into a functional e-commerce website.
Six assignments compose the homework portion of the class, and these assignments are primarily responsible for demonstrating student mastery of material presented in the course lectures and virtual labs. These homeworks typically span multiple weeks, and we allow them to be completed either by individuals or by teams of two. Course feedback strongly confirmed that students prefer a series of integrated assignments that work toward a complete e-commerce site rather than independent exercises completed in isolation. In addition to the homework assignments, the course project allows students to demonstrate their understanding of the underlying concepts. Several of these course projects have transitioned into successful e-businesses.

Nevertheless, we have discovered a consistent shortcoming in the area of security. Even when specifically including security topics in lectures, we find that students fail to incorporate this material in their course projects. Students neither apply coding techniques that prevent common attacks nor use the security tools that are provided as part of our computing infrastructure.

In regards to the first flaw, SQL injection is one of the most common attack vectors across all software systems accounting for 14% of flaws targeted by attackers in 2006 [3]. While class lectures have covered this attack vector even to the depth of a live demonstration of an SQL injection attack against a website mockup, students still frequently omit sanitizing user input. This trend suggests that lectures alone fail to provide sufficient motivation for security. We believe that the homework assignments themselves should provide students with a hands-on learning experience so they fully understand the dangers of lax security.

Beyond this first deficiency, we have found that students do not apply the provided resources for securing their websites. Our classroom setting provides a common (%NIX) web server cluster for students. Because the web servers are a shared resource, the default permissions often provide a means for discovering private information, e.g., database passwords. Students routinely fail to secure their web directories and files so that they are only accessible to themselves and the web server and not all other system users. The suPHP extension,1 which allows PHP scripts to be read and to be executed only by their owners, is installed on the web server, but students do not take advantage of this ability. It is not uncommon to discover private information simply by reading the PHP source code from an interactive shell.

In part, student failures may only be indicative of the abstract nature of the presentation and a failure to observe these phenomena in the real world. After all, it is well-established that programmers frequently make mistakes in infrequently executed code [11]; perhaps students lack the motivation (expressed via concrete test cases) to invest their time and efforts in website security, which is frequently not considered until after a data breach. Students themselves have picked up on this deficiency (e.g., covering cryptograpth in the abstract) by critiquing the fact that the e-stores that are created during the existing homework assignments do not implement any security, making them inadequate training for the real world. Thus, our experience suggests security topics should be incorporated into the course not only through lectures but also though the hands-on homework assignments.

1http://suphp.org/

1.2 Contributions

Our goal is to integrate security into the course material so security topics are covered in lecture and homeworks in tandem. The correct time to emphasize proper coding practices is while initially learning a language, not included as an afterthought after other topics are exhausted. Common errors and mistakes should be guarded against by explicitly providing working examples (preferably with the students’ own code) of possible threats and by giving students the opportunity (or requiring them) to correct any such problems that are revealed. In section 2, we present a case study that builds on the success of virtual labs to create a series of homework assignments designed to introduce web technologies and cover the various security topics associated with each. Section 3 covers our testing methodology for identifying security vulnerabilities and for requiring students to correct any errors revealed by the test suites.

2. ARTIST EXCHANGE CASE STUDY

One of the challenges presented in teaching an e-commerce course is how best to present the myriad of available web technologies. Students taking the course are typically third or fourth year undergraduates with a considerable amount of programming experience, typically in Java. Unfortunately, Java is not one of the languages traditionally associated with web programming or e-commerce software. While there are Java-based technologies for creating dynamic web pages, they do not enjoy the prominence of the Linux, Apache, MySQL, and Perl / PHP / Python (LAMP) software stack. The latter scripting languages are almost universally supported on web servers whereas the former solutions typically require special resources making them ill-suited for a course designed to garner practical experience for students.

The Artist eXchange case study is patterned after wildly successful social networking sites that students are likely to use daily (e.g., Facebook or MySpace). The website provides a location for musicians to create an artist profile, uploading pictures and musical samples of their work. Users have the ability to search for artists and rate the music samples so over time each artist receives an electronic reputation for the quality of his or her music. Our reference implementation is programmed using a variety of the major web technologies (HTML, CSS, JavaScript, and PHP to name a few), but there are no requirements that mandate these technologies. We simply believe that the predominance of the LAMP software stack makes its solutions preferable as part of a general introduction to web programming.

The pedagogical goal of the Artist eXchange website is to familiarize students with languages that may not have been previously encountered, not to introduce programming. Thus, the approach taken in lab write-ups is to introduce a topic, provide a code snippet that illustrates how to use it, and ask students to demonstrate their understanding by incorporating a similar feature in a different location on the website. For example, the HTML DOM is introduced via form validation with JavaScript. Students see a code fragment that verifies that a password and password confirmation input both contain the same value and that displays an alert if the values do not match. Once the student has integrated the code fragment into his or her own website, the lab follows up with a series of additional exercises which increase in complexity. For example,
code provides little beyond the names of files and directories. By constructing the website from the ground up, students have an invested interest in proper security practices. Moreover, integrating security topics in the individual lab assignments allows students to understand the “defense in depth” aspect of security.

Finally, the case study itself reflects an opinion that has been clearly expressed by students: homework assignments should not be independent throwaway exercises but should instead build upon each other. By coalescing the flexibility provided by virtual labs with the in-depth coverage demanded by existing homework assignments, we hope to combine the best aspects of each while reducing the workload on students. Moreover, we hope that modeling the case study around popular social networking sites will enable students to see the practical application and social value of the assignment. Both Teague [6] and Treu and Skinner [7] state that women in particular become more interested once they understand a project’s personal or societal relevance. The Artist eXchange website also provides an ideal starting place for discussing computing ethics (simply consider that the website potentially allows users to download an artist’s work gratis), which—as stated in the introduction—is itself an important part of teaching security in the classroom.

3. AUTOMATED TESTING

Accompanying the Artist eXchange website is an extensive suite of automated tests that verify that each task is completed correctly. We deem an automated testing framework essential for a number of reasons. First, it enables students to receive real-time feedback on the quality of their work. This feedback comes before any assignment is graded by the instructor, which allows students to correct errors before turning their attention to the next task. The immediate feedback is critical, for each assignment builds upon the previous one. Any errors must be corrected before moving on to the next task. Second, as security topics are introduced, it is essential that the existing services provided by the website not be impacted. A student could easily make a site resilient to SQL injection attacks by removing all database-related code. This “solution” clearly contradicts the goal of the exercise and is expressly guarded against by the myriad of functionality tests. Regression testing also gives students confidence that the modifications required by the lab exercises are completed correctly. Finally, today’s websites are extraordinarily difficult to test by hand. Web applications combine client-side and server-side scripting and checking one frequently involves disabling the other, a process impractical when grading between 50 and 100 individual assignments. Fortunately, there are frameworks that allow both to be verified in an automated fashion.

The TestNG framework [2] was chosen for constructing the Artist eXchange test suites for two major reasons. First, TestNG is not a simple unit-testing framework—it is designed to cover all categories of tests, including unit, functional, and integration testing. Because most of the development work that would traditionally be covered by unit testing is included in server-side scripts, the test suites cannot test this functionality directly. Instead, most test cases cover an entire script (e.g., the content of a page or authorization via the websites login forms), which hampers the use of traditional unit-testing frameworks. Second, TestNG supports the notion of test dependencies, which are pervasive and are not typically covered by traditional unit-testing frameworks. In addition, TestNG supports regression testing (the ability to verify that existing functionality is not broken by new changes). This feature is essential for an automated testing framework, as it allows students to verify that new changes do not break existing functionality.

Figure 1: A screenshot of an Artist eXchange profile. The musician has uploaded a photograph and a music sample, and users may submit feedback regarding their opinion of the music.

1. Clearing the contents of the password and password confirmation input when the values do not match
2. Checking that all of the required fields have received values
3. Verifying that an email address is valid by ensuring that it conforms to a regular expression

Our intent is to provide a ladder of exercises for students to demonstrate their mastery of the material, an approach that has been well-received in the course’s existing virtual labs.

The scope of the Artist eXchange website is kept intentionally small for two reasons. First, we recognize that many students may already have significant experience in web development. These students should not be overly burdened by requiring them to conform to a specification instead of learning new material. In addition, the case study itself is not the course project: the case study only serves as a means to learn popular web technologies. Students should focus their time and energy on a project that is of interest to them. Second, a minimal scope allows the Artist eXchange website itself to be modular. If additional activities are desired on any particular technology (e.g., ASP and .NET or Flash), it is trivial to add this technology to the existing site. Hence, the lab activities can adapt to student interests with whatever level of coverage is deemed appropriate by the instructor.

Because one of the major goals of the case study is web application security, the labs work toward correcting common vulnerabilities found in websites, particularly SQL injection vulnerabilities. The Artist eXchange website is constructed from a minimal code base provided to students; the skeleton
sive in web application tests. Most of the test cases depend on the successful completion of other tests. For example, if the login form prompts for an email address and password instead of a username and password, none of the tests that depend on the presence of the latter elements should be executed. TestNG’s dependencies allow these complex relationships to be stated, and a failure in any dependency results in the dependent test cases being skipped instead of failed.

The test cases themselves make extensive use of the Selenium framework [4] for web application testing. Selenium allows static HTML as well as dynamic client-side and server-side scripts to be tested. Developers (students) run a local Selenium RC (remote control) server on their machine, and this server actually executes each test. The Selenium RC server enables tests to be run using a variety of browsers (e.g., Firefox, Internet Explorer, Chrome, Safari, and Opera), which gives the tests additional flexibility. Selenium RC discourages students from coding only to pass the test case in two ways. First, the cost for each test execution is relatively high when compared to traditional unit tests. Creating new browser sessions means that the complete test suite requires between 30 seconds and 5 minutes to complete, instead of the few seconds required in many unit testing scenarios. Second, each new browser session appears on the student’s computer; the pop-ups make it difficult to run the tests mindlessly in the background. In addition, by executing each test on the student’s machine, the cost of the test in terms of server resources (CPU and memory) is minimized. The cost is born almost completely by the developer.

The combination of TestNG and Selenium again facilitates the modularity goal of the case study. Should an instructor choose to add additional assignments that cover different web technologies, it is simple to incorporate additional test cases for these exercises.

Automated Testing Example

Students start by logging in to the virtual labs website, which contains a list of lab activities. A student selects a specific lab; the website then prompts for the location of the student’s code that implements the lab specifications. Once this information is provided, the student may execute the lab’s test cases.

The test cases for a lab are typically grouped by specification. For example, the first lab exercise includes test cases for static HTML content and client-side scripts performing form validation. Students select the test cases they wish to execute and click the start button (figure 2). The page displays the status of each test case as soon as it finishes execution. Following completion of the entire test suite (the exact test cases included in the suite are determined by the student’s selection), a detailed report is generated.

Figure 3 displays an example report that is automatically created after completing a test suite. The report homepage contains a pie chart of the number of test cases passed, failed, and skipped (top of figure 3). A test case may be skipped if it requires the successful completion of another test case; when the other test case fails, it is not sensible to execute the dependent test. For example, the login test cases are divided into checking the login form’s input elements and actually attempting to log into the website using existing login credentials. When the login form does not contain all of the required elements, the attempt to log in is skipped—if it was executed, it would fail because an element on the login page is not present. The report provides detailed error messages regarding the exact failure (bottom of figure 3). A missing username input generates the error message “Missing username input ("//form[@id='loginForm']/input[@id='usernameInput']").” which not only specifies the missing element but also identifies the XPath expression used to find the element on the page. The XPath expression is particularly helpful for identifying when an element is present but one of the attributes required by the specification (e.g., an id) is missing or incorrect.

After reading the report to identify any implementation mistakes, the student makes the necessary corrections and may execute the test suite again. We do not limit the number of times the test suite may be executed because 1) each test is actually executed on the student’s machine (which minimizes server load) and 2) the repeated feedback can help students implement all of the specifications. Once all of the test cases are passed, students are assured that their implementation meets the requirements outlined in the lab assignment.

4. CONCLUSION

We have developed an innovative series of assignments that merge the best aspects of an existing course’s virtual labs and homework exercises. The Artist eXchange case study provides an introduction to web technologies, and its automated tests allow students to view their progress toward completing lab objectives in real time. Our Fall 2009 offering of Electronic Commerce Technologies will be the first to incorporate this work. Hence, we will solicit student feedback throughout the process to gather student comments, complaints, and suggestions.

Based on previous feedback, we expect students to appre-
Figure 3: A detailed report generated after executing a series of test cases. The pie chart (top) presents a quick overview on the percentage of test cases passed, failed, and skipped (shown in the colors green, red, and yellow respectively). Clicking on the test provides a more detailed report (bottom) that indicates the cause of failure for each failed test case.

ciate the virtual nature of the assignments since it does not require attendance at fixed lab times. In addition, we think that students will be interested in incorporating other web technologies into the case study—e.g., Flash, ASP, or the .NET framework. The hands-on, practical nature of each assignment should make the experience more exciting to students in general and particularly to women in computing. Finally, by integrating security topics into the assignments, we hope to encourage students to consider this vital aspect of computer science and to incorporate it into their future projects.

5. ACKNOWLEDGMENTS
The authors gratefully acknowledge this project’s generous support provided by the National Science Foundation’s funding of proposal 0717739, “Secure eCommerce: A Modular Course Supported by Virtual Laboratories.”

6. REFERENCES