Proceedings of the 7th World Conference on Information Security Education
9-10 June 2011, Lucerne, Switzerland

Creating Shareable Security Modules

Kara Nance[1], Blair Taylor[2], Ronald Dodge[3], and Brian Hay[4]

[1,4] University of Alaska Fairbanks, Fairbanks, AK, USA
{klnance@alaska.edu, brian.hay@alaska.edu}
[2]Towson University, Towson, MD, USA
btaylor@towson.edu
[3]United States Military Academy, West Point NY, USA
ronald.dodge@usma.edu

Abstract. While there is increased appreciation of the need to provide students with education in computer security, there are significant challenges associated with the creation of shareable computer security modules that can be used by a wide-range of educators. This paper discusses some of the challenges that educators currently face in this area, and presents a means to couple a successful framework and infrastructure environment to address some of the associated issues. Two examples are provided that link the framework and infrastructure followed by suggestions for future research and development in this area.

Keywords: Computer Security Education, Computer Education, Repositories.

1 Introduction

This paper discusses a framework for creating shareable security modules for information assurance. We explain the environmental and pedagogical challenges associated with the creation and sharing of educational resources, followed by a description of two successful projects that can be coupled to facilitate the process - the Security Injections@Towson (SI@T) and the Remotely Accessible Virtualized Environments (RAVE) projects. We provide example labs that demonstrate the coupled framework in action and conclude with future considerations that will facilitate the creation and continued evolution of shareable security modules.

2 Challenges

The benefits to a hands-on learning environment are widely recognized. Access to hands-on environments not only strongly reinforces lecture concepts, but also allows students to experiment with and extend concepts presented in the classroom. However, there is a great deal of effort required for individual instructors to create educational materials that extend course concepts to a hands-on learning environment. The challenges and time requirements can make the exercise prohibitive for many educators. Moreover, when those efforts are successful, there is often not an easy way to leverage the effort of any given teacher to improve the capabilities of the entire community.

2.1 Environmental Challenges

Below is a list of questions instructors may need to address when creating a hands-on computer lab experience:

- Do all of the students have the same configuration?
- Do the students all have the same computing platform?
- Do they all have the same operating system?
- Do their machines have enough resources to run the lab exercise?
- How do I know that they all started from the same configuration?
- If I am not sure that they all started from the same configuration, how can I grade them appropriately?
- When a student has a problem with the lab exercise, how can I provide help to them?
- If I need to make a change to the lab exercise or configuration, how do I distribute that to all students?
- If I am not at my own computer or at the school, how can I work on the lab exercises? [6]

In addition to configuration issues, the instructor needs to worry about student access to the lab resources, load balancing among limited resources (such as software licenses), and managing the instructor time so that individual student needs can be met. While these issues are complicated in a face-to-face laboratory environment, they become even more challenging when the educational environment involves distance education or even an asynchronous local experience.

When the topic being taught is computer security, additional issues arise as the hands-on labs and activities frequently can only be done in an isolated environment. Studying issues such as the malware behavior and cyberdefense exercises would not be safe (or in some cases, legal) on production networks. Yet, the ability to gain hands-on experience with the computer security concepts presented throughout the curriculum is essential if we want students to be able to address the evolving security needs of the nation.

2.2 Pedagogical Challenges

In addition to the environmental challenges faced by instructors, further work is required to produce materials which support a meaningful hands-on educational experience for the student. This includes providing adequate foundational elements to bring all students to a common level, educational content to meet the learning objectives, reflective activities to ensure that the learning objectives have been met, and extension activities to demonstrate how the concepts fit into the big picture. In addition, the current ad hoc nature of most Computer Science (CS) labs inadequately address synthetic and analytical thinking, Most programming labs are structured towards the goal of "getting the code to run." For example, a lab assignment which requires students to compute the average of three test scores could be tested with the input values of 100, 100, and 100; and submitted with an answer of 233.33. In many cases, students are so relieved that the code ran they give little thought regarding the reasonableness of the answer. To address these challenges and better prepare students as security professionals, there have been increased efforts towards creating information assurance laboratories [1-3]. However, while more instructors recognize the need for incorporating security into the curriculum, many are hindered by the environmental challenges listed above, resource limitations, time constraints, insufficient security training, and lack of effective pedagogical materials.

3 Framework for Security Modules

Based on our own experiences with these challenges, we propose guidelines for creating information assurance resources. Specifically, a framework for shareable security modules should:

1) be broadly applicable across institutions and courses
2) be extendible to meet the needs of diverse audiences
3) be easy to use from a student perspective
4) be easy to identify, access, and implement for instructors
5) encourage active learning
6) facilitate and stimulate development of new modules
7) be largely platform independent

The combination of two successful NSF research projects provides an exciting opportunity to begin to meet these important guidelines. The following sections describe the Security Injections@Towson (SI@T) (NSF Project 0817267) and the Remotely Accessible Virtualized Environments (RAVE) (NSF Project 0123152) and provide two examples of how the project outcomes can be utilized in tandem to begin to meet the requirements for a framework for shareable security modules.

3.1 Security Injections@Towson

For the past five years, researchers at Towson University have worked with instructors across five diverse institutions, to incorporate security into the CS curriculum. The project has targeted the introductory programming courses required of all CS majors: Computer Science I (CS1), Computer Science II (CS2), and the preparatory course in programming logic (CS0); as well as the Computer Literacy course offered to non-majors. The goals of the project were to 1) increase faculty awareness of secure coding concepts 2) increase students' awareness of secure coding issues 3) increase students' ability to apply secure coding principles and 4) increase the number of security-aware students. Towards this end, they developed and implemented a series of self-contained security injection modules that target key security concepts including integer overflow, buffer overflow, and input validation for the programming courses and phishing, passwords, and cryptography for the literacy course [1].

The process for material development began with an initial set of draft modules that were piloted in local classrooms. To encourage collaboration, researchers held on-site faculty workshops at each of the participating institutions, using the modules as starting points for discussion and review. Revised modules were deployed in a variety of educational contexts. Formal assessment included pre and post-tests, code checks, and faculty surveys to identify factors that worked well across different demographic groups. Feedback from workshop participants, assessment results, and advice from an expert evaluator, shaped the formation of the resulting security injection modules.

The format for the security injection module includes four components as described below:

Background: The purpose of this section is to set the context of the assignment, provide necessary background information for the security lab, and motivate students for future learning. This section includes a brief summary of the targeted security issue, a description of the problem and risk, code snippets which demonstrate the vulnerability, and real-life examples which describe actual occurrences of security incidents that have been documented in the news or other media and are selected to peak students' interest and motivate them to fully understand the concept.

Real-life Example: In December 2005, a Japanese securities trader made a $1 billion typing error, when he mistakenly sold 600,000 shares of stock at 1 yen each instead of selling one share for 600,000 yen. A few lines of code may have averted this error. [7]

Problem-Security-Related Lab: Creating interesting and relevant CS lab assignments has always been challenging. Students today are genuinely interested in security; therefore security-centric labs not only teach important security concepts but can help increase interest and motivation. Dovetailing the traditional CS core concepts with relevant security topics – arrays with buffer overflow, data types with integer errors – has been effective. A model for mapping security topics to primitives, courses - CS0, CS1, or CS2, and learning objectives [4] is easily expanded to other courses.

Checklist: Security checklists, which target a security vulnerability or topic, have been developed using feedback from students, instructors, assessment, and an expert evaluator. Checklists help students check their code and simultaneously learn and internalize important security concepts. Checklists provide a quantifiable list of security criteria to aid in writing secure code and further reinforce security principles by encouraging self-evaluation and learning reinforcement. The checklist can also be used for peer reviews, collaborative learning, and assessment. Repeated exposure to the checklists and security concepts facilitates use of the checklists and reinforces the security principles. Additionally, as the process of using the checklist becomes routine, the expectation is that students will practice this habit as programmers.

Analysis- Discussion questions: Discussion questions require students to analyze and summarize their results and promote critical thinking, analysis, and reflection. Additionally, students' answers to these questions provide valuable and immediate feedback to the instructor.

The template for the security injection modules was created with an eye towards the learning sciences and borrowing from the more structured laboratory approach employed by the traditional sciences such as biology and chemistry. By motivating students with background information, including self-checks and reflective questions, students are encouraged to analyze the process, the results, and the security implications. The use of a standardized lab format was also found to be beneficial to both students and instructors. Students gained familiarity with the structure and process for completing each assignment. Instructors could pick and choose parts of the labs for inclusion in their own assignments and most importantly, this model proved easy to expand for new courses and new topics.

3.2 RAVE

One common barrier to the utility and adoption of a lab repository is the heavy dependence on infrastructure and support. These requirements include specific hardware and software requirements for the labs, shared computer labs with a fixed number of computers, and to the system administration of the lab facilities.

The model implemented by the RAVE (Remote Access Virtual Environment) project, creates shareable virtual environments built to support many institutions remotely accessing a set of resource clusters. It replicates and builds on the successful prototype ASSERT Lab at the University of Alaska Fairbanks [8]. Many papers have discussed the benefit of virtualization supporting information assurance laboratory exercises. The advantage of RAVE comes from the nature of a shared set of computing recourses; one virtualization resource center is used by many different institutions. The RAVE architecture consists of multiple virtual resource centers. Combining the shareable lab exercises with a standard lab infrastructure removes a significant barrier limiting many institutions.

4 Examples

The two NSF-funded projects described above can provide a catalyst for the creation of shareable computer security modules. In order to demonstrate this concept, two examples are provided. The first takes an existing example from the SI@T suite of exercises and demonstrates how coupling the exercise with the RAVE environment will address most of the challenges identified in section 2. The second exercise takes an existing RAVE scenario and builds an associated educational component using the SI@T framework.

4.1 Example 1 – SI@T modules in the RAVE environment

The SI@T Project has resulted in a collection of valuable resources in a common framework easily utilized by instructors. While many instructors have an infrastructure in place in which students can conduct these exercises, most institutions suffer from some (if not all) of the environmental challenges listed in section 2. As described in section 3, the exercises consist of four sections. Three of the four sections are self-contained. Combining the hands-on or problem section with the RAVE capabilities alleviates all of the environmental challenges listed in section 2.

After completing the background section, the student is given access to a RAVE environment in which to test the applied component. The nine identified issues are addressed through the configuration of the RAVE environment.

- Issues 1-5 have to do with student resources configuration. In this case, all students are given identical virtualization environments; so system components, platform, operating system, machine resources, and starting configuration are all ensured to be uniform.
- Issue 6, handling non-identical configurations, is no longer an issue since all environments are homogeneous.
- Issue 7 is concerned with student assistance. RAVE provides several capabilities to assist in this area including remote access by instructors, permissions to view and assist student accounts and snapshot capabilities to further interact with students.
- Issue 8 is concerned with changes to the configuration and distributing changes. Since RAVE is a virtualized environment and images are created on demand, this issue is easily solved.
- Issue 9 has to do with student accessibility to the environment to complete the hands-on component. Since RAVE environments are remotely accessible and available around the clock, students are free to complete the exercises within the constraints of the timeline required by the instructor.

Thus, using RAVE as an environment for completing the hands-on component of the SI@T suite of exercises addresses many of the identified challenges associated with hands-on computer security labs.

4.2 Example 2 – RAVE exercise using the SI@T framework

As IA course offerings at colleges and universities have increased dramatically over the last 10 years, many institutions have struggled with the issues identified in section 2.1. In the previous section, we identified how current IA modules from the SI@T benefit from utilizing the infrastructure provided by RAVE. Outside of the fiscal and resource management benefits of leveraging the RAVE virtualization resources center model, the greatest curricular enhancement comes from instructors being able to now more easily share IA lab exercises. An enhancement to provide a more reusable set of exercises is to take existing scenarios developed by faculty and rewrite them adopting the framework discussed in section 3.

As an example of this process, we rewrote a lab exercise (described in [9]) instrumented in a RAVE cluster to follow the SI@T model. The module was originally written to permit other faculty to adopt the module for use in their own institutions. Given the variability in infrastructures, 70% of the module focused on how to set up the hardware and software, rather than the learning objectives. The four sections in the new model broke the content into a format that was more easily followed by students.

The *background* section was mostly present in the existing module, however, adding details that tied the objective to a real world event provided more motivation to the students. The second section, *Problem - Security-Related Lab,* was directly ported over. The third section, *Checklist*, required iterating through the lab exercise, identifying key components that tied to the learning objectives, providing a guided walk through of essential concepts of the security topic. The final section, *Analysis-Discussion questions*, was already present in the original lab exercise.

The process of migrating the original lab construction to the SI@T model resulted in an exercise that provides the advantages listed in section 3, providing a more shareable module that can be accessed virtually anywhere in the world.

5 Future Considerations

The framework for the computer security modules described in the previous section, builds on successful NSF research efforts to provide meaningful, hand-on exercises that address many of the identified environmental and pedagogical challenges. What remains is the identification or creation of a repository environment to facilitate the sharing of the modules on a much wider scale. There are currently several efforts underway that are addressing these challenges and coupling the results of this research effort with successful repository development will be essential to ensure that the research efforts are leveraged to minimize duplication of effort and maximize the sharing of resources. The repository must allow instructors to adapt modules to their own environment and then contribute these newly evolved modules back to the repository for others to use. It needs to be scalable so that the topical areas can expand as the technologies to support other areas are identified and utilized.

While there are several repository efforts underway for IA educators, the repository frameworks are each unique, limiting the ability for instructors to locate exercise labs that support their curriculum and infrastructure. The NSF-funded Ensemble Project [5] may be a candidate for a repository, or at least provide a foundational framework to guide the development of a more specialized framework. The project uses a distributed portal approach intended to coordinate across communities. A second related NSF-funded project (# 0231122 and 0618680) is SEED: A Suite of Instructional Laboratories for Computer SEcurity EDucation [3] which has a growing suite of complete educational laboratory experiences, provides a wealth of experience in the development and deployment of security education modules, but is also in search of a repository. Likewise, PRISM: A Public Repository for Information Security Material [2], provides a repository framework that should be further evaluated as a repository environment for this framework.

While efforts continue, much work remains to be done in order to reach the ambitious framework goals outlined in this paper.

Acknowledgements: This research has been funded in part by the National Science Foundation through the Division of Undergraduate Education (#1023125) and (#0817267).

References

1. Security Injections @ Towson University. Available from: http://triton.towson.edu/~cssecinj/secinj/.
2. Garramone, V. and D. Schweitzer, "*PRISM: A Public Repository for Information Security Material*," in Colloquium for Information Systems Security Education(CISSE*)*. 2010: Baltimore, MD.
3. Du, W. and Wang, R.,. "*SEED: A Suite of Instructional Laboratories for Computer Security Education (Extended Version),"*. In The ACM Journal on Educational Resources in Computing (JERIC), Volume 8, Issue 1, March 2008.
4. Taylor, B. and S. Azadegan, "*Moving Beyond Security Tracks: Integrating Security in CS0 and CS1*," in Technical Symposium on Computer Science Education (SIGCSE), ACM, Editor. 2008, ACM.
5. Hislop, G.W., et al., "*Ensemble: creating a national digital library for computing education*," in Proceedings of the 10th ACM conference on SIG-information technology education. 2009, ACM: Fairfax, Virginia, USA. p. 200.
6. Nance K. and V. Nestler. Unpublished manuscript. 2009.
7. Costello, Miles. "*Fat fingered typing costs a trader's bosses £128m*," The Times Online, December 09, 2005
8. Hay, B. and K. Nance. "*Evolution of the ASSERT Computer Security Lab"*. 10th Colloquium for Information Systems Security Education. Adelphi, MD. June 2006.
9. Hay, B., Dodge, R., Nance, K., "*Using Virtualization to Create and Deploy Computer Security Lab Exercises,*" proceedings of the 23rd International Information Security Conference (SEC 2008), 8 – 10 Sept, 2008, Milan, Italy