Enhancing Security Education with Hands-on Laboratory Exercises

Wenliang Du
Syracuse University
wedu@ecs.syr.edu

Karthick Jayaraman
Syracuse University
kjayaram@syr.edu

Noreen B. Gaubatz
Syracuse University
nbgaubat@syr.edu

Abstract—**We present an instructional suite of labs that can help in teaching security courses. Our labs are publicly available for use. We have used these labs in teaching security courses and also helped several other instructors adopt these labs to suit their style of teaching. Based on this experience, we present some guidelines for using these labs in security courses. Furthermore, we also describe the results of using these labs among students.**

I. Introduction

It is widely known that learning-by-doing can significantly enhance student's learning in many disciplines [1], [5], including computer security education [3]. Eight years ago, when we started looking for well-design hands-on projects for our security courses, we could not find many. Although we did find a small number of good individual laboratories [2], [4], [6], there were two problems in using them. First, the coverage of security principles was quiet narrow. Second, the lab environments were varied and as we intend to use these labs in a single course, students would have to a spend significant amount of time to learn to use the underlying environment.

Motived by the need for better, coherently-designed, and well-supported lab exercises for security education, we started our journeys in 2002. Our objective was to develop a suite of labs that cover a wide spectrum of principles, ideas, and technologies that are essential for teaching computer security. We call these labs the SEED labs (SEED stands for SEcurity EDucation). Our design particularly focused on two aspects: the lab environment and the lab contents.

Low-cost and consistent environment: First, we wanted to have a unified lab environment that is consistent for different lab exercises, so students do not need to learn a new lab environment for each lab, because learning a new environment is not easy. Second, the environment used for these labs must be affordable to enable wider adoption. With these constraints in mind, created a lab environment comprising the open-source Linux Operating system and a number of open-source software tools. The lab environment can be created on student's personal computers or department's general computers with the use of a virtual machine (VM) software. There are free VM software, such as VirtualBox, and even the commercial product VMware is free for educational use through VMware's academic program.

Lab contents with wide coverage: Unlike some traditional courses, such as operating systems and compilers, there is no common consensus on how security should be taught, and what contents should be covered. Therefore, we decided not to base our labs on a particular course. Instead, we decided to develop labs that can cover a wide spectrum of security principles. Although the style and focus of security courses could vary with instructors, the fundamental security principles that all instructors would like to teach do not significantly vary. Because the labs cover a range of security principles, instructors could choose the labs from our list to fit their course based on the security principles they want to cover. Adoption of our lab exercises do not require any changes in the course structure.

After 8 years' working on the project (funded by two NSF grants), we have developed over 25 labs, most of which have been used and tested by courses at our own university; several of them were also used by instructors from other universities. The labs are available from our project web site http://www.cis.syr.edu/~wedu/seed/. Since 2007, the labs have been downloaded for over $40,000$ times from the computers outside of our university network. The labs are broadly categorized into three classes:

1) **Vulnerability and attack exercises:** The objective of these exercises is to illustrate and help the student learn a vulnerability in detail. Each of these exercises typically has four parts. First, students will understand the vulnerability based on simple examples and identify them in either a real program or a synthetic example. Second, students will construct real attacks to take advantage of the vulnerability. Third, students will fix the vulnerabilities and also comment on prevailing mitigation methods and their effectiveness.
2) **Design and implementation exercises:** The objective of these exercises is to help the student understand the concepts of secure system development. Students are expected to design a system that provides a specific security functionality. The exercise outlines clear objectives and students would explore various choices for fulfilling the objectives, analyze the impact of the choices on security and performance, make appropriate design decisions, and finally implement and demonstrate their system.
3) **Exploration exercises:** The objective of these exercises is to coerce the students to explore or play with an existing security functionality. Exploration labs are like a "guided tour" of a system, in which, students can

"touch" and "interact with" the key components of a security system to either learn or question the underlying security principles that the system is based on.

Experiments and Evaluation: Over the last 7 years, we used the SEED labs in two graduate and one undergraduate security courses. We have been revising those labs based on student feedbacks. Now, most of the labs are quite stabilized, and are ready for use by other instructors. So far, fifteen other universities have informed us that they have used some of our SEED labs in their courses. To evaluate the effectiveness of our labs, we collected survey data from students for each of the labs they used. The results are quite encouraging. We will present our evaluation results in this paper.

II. Description of Labs

In this section, we provide a brief description of our SEED labs. Since the actual lab description for each lab ranges between 2 pages to 10 pages, we cannot include them in this paper. We will only describe a selected few. Readers can find the detailed lab descriptions from our web page http://www.cis.syr.edu/~wedu/seed/. Table I contains the list of labs.

A. Vulnerability and Attacks Labs

All the vulnerability and attack labs are structured to let the student learn the vulnerability in detail by creating an exploit for a program with known vulnerabilities. Furthermore, the students are asked to comment on the prevailing mitigating methods and their effectiveness. Table I contains all 12 vulnerability and attack labs. Broadly, we have three categories of vulnerabilities, general software vulnerabilities, network protocol vulnerabilities, and web application vulnerabilities. We describe the nature of the labs in each of the categories. The detailed lab description can be obtained from our web site. From our experience, the vulnerability labs are heavily used by several instructors.

- *General software vulnerabilities:* We have lab exercises focusing on general software vulnerabilities such as buffer overflows, format-string vulnerabilities, race conditions, etc. Latest versions of `Linux`-based systems have several features for countering these vulnerabilities. Therefore, where necessary, the lab exercises provide information on how to disable the vulnerabilities in order to perform the lab exercises.
- *Network protocol vulnerabilities:* In this category, we have two labs, namely TCP/IP attacks and DNS pharming. In the TCP/IP attack lab, students discover and exploit vulnerabilities of the TCP/IP implementations. In the DNS pharming lab, students understand how DNS pharming attacks could be used to manipulate the name resolution process of trusted web sites to compromise the user privacy.
- *Web application vulnerabilities:* We have labs focusing on web-based vulnerabilities such as cross-site scripting, cross-site-request forgery, clickjacking, etc. The lab exercises for these labs involve exploiting a web application with known vulnerabilities. We have setup some web applications inside our pre-configured virtual machine that students will use for these labs. Therefore, instructors need not maintain any web servers or install or configure applications for using these labs.

B. Exploration Labs

In the exploration labs, students both learn and critique the design and implementation of an existing security component. We have four labs in this category:

- *Packet sniffing and spoofing lab:* In this lab, students learn how packet sniffing and spoofing tools are implemented. Students will start with some simple tools, understand them by reading their source code and using them, and modify them to implement some interesting variants.
- `Linux` *capability lab:* The objective of this lab is to train students to use the capability system implemented in `Linux` and also gain insights on how the capability system is implemented in the operating system.
- *Web browser access control lab:* The objective of this lab is to help the student understand and critique the same-origin policy commonly implemented in all web browsers for access control.
- *Pluggable authentication module:* The objective of this lab is to illustrate the use of the flexible authentication technique called pluggable authentication module (PAM) implemented in `Linux`. PAM provides a way for developing programs that are independent of the authentication scheme.

C. Design Labs

Each of the design labs provides clear functional requirements for implementing a system. Furthermore, students are also provided supporting material and documents for helping them. To help the students get started, some design choices are outlined. The students are first asked to provide a detailed design of the system describing why they choose particular design choices and how they impact the security and performance of the system. The faculty or TA could provide feedback on their design, before the student implement the system. Currently, we have nine design labs. We provide a brief description of these labs as follows:

- `Linux` *Firewall Lab and* `Minix` *Firewall Lab:* In this lab exercise, students are expected to implement a simple packet filtering firewall. The firewall should provide an interface to the user to enable configuration. Each firewall rule describes the types of packets, in terms of their properties such as source IP address, protocol, etc., that will be allowed or blocked. The design labs are available in both `Linux` and `Minix` operating systems. For `Minix`, students have to modify the `Minix` kernel to implement the firewall. For `Linux` operating system, students will use the kernel loadable module mechanism and the `netfilter` hooks to implement the firewall.
- *IPSec Lab:* In this lab, students will implement a simplified version of the IPSec protocol for the `Minix`

Types	Labs	Security principles	ST	NW	PG	SWE
Vulnerability and attack labs	Buffer-overflow Vulnerability Lab	SC	UG		UG	UG
	Return-to-libc Attack Lab	SC	UG		UG	UG
	Race-Condition Vulnerability Lab	SC	UG		UG	UG
	Format-String Vulnerability Lab	SC	UG		UG	UG
	Chroot Sandbox Vulnerability Lab	AC, SD	UG			UG
	TCP/IP Attack Lab	SC, SD		UG	UG	UG
	DNS Pharming Attack Lab	SD		UG		UG
	Cross-Site Scripting (XSS) Attack Lab	SD		UG	UG	UG
	Cross-Site Request Forgery (CSRF) Attack Lab	AC, SD		UG	UG	UG
	ClickJacking Attack Lab	AC, SD		UG	UG	UG
	SQL Injection Attack Lab	AC, SC, SD		UG	UG	UG
	Set-UID Program Vulnerability Lab	AC	UG		UG	
Exploration labs	Pluggable Authentication Modules Lab	AU	UG			UG
	Packet Sniffing and Spoofing Lab	AU	UG	UG		
	Linux Capability Lab	AC	UG			UG
	SYN-Cookie Lab	CG		UG		
	Secret Key Encryption Lab	CG	UG			
	One-way Hash Function Lab	CG	UG			
	Public-Key Cryptography Lab	CG	UG			
	Web Access Control Lab	AC, SD	UG	UG		
Design labs	Set-RandomUID Sandbox Lab	AC	G			G
	Minix Capability Lab	AC, SD	G			G
	Minix Role-Based Access Control Lab	AC, SD	G			G
	Encrypted File System Lab	CG, SD	G			G
	Address Space Layout Randomization Lab	SD	G			
	IPSec Lab	CG, SD	G	G		G
	VPN Lab	CG, SD	G	G		G
	Linux Firewall Lab	AC, SD	G	G		G
	Minix Firewall Lab	AC, SD	G	G		G

Legend

ST - Systems, **NW** - Networking, **PG** - Programming, **SWE** - Software engineering.
G - For Graduate students only, UG - For both Undergraduate and Graduates.
AU - Authentication, AC - Access control, CG - Cryptography SC - Secure coding, SD - Secure design.

TABLE I
THE COVERAGE OF SEED LABS

operating system. The lab tasks have been simplified to facilitate the students to complete the lab exercise within four to six weeks. Students need to use their IPSec implementation to construct a Virtual Private Network (VPN).

- *VPN Lab:* VPN is a widely used security technology. VPN can be built upon IPSec or Secure Socket Layer (SSL). These are two fundamentally different approaches for building VPNs. In this lab, we focus on the SSL-based VPNs. This type of VPN is often referred to as SSL VPNs. The learning objective of this lab is for students to master the network and security technologies underlying SSL VPNs. The design and implementation of SSL VPNs exemplify a number of security principles and technologies, including cryptography, integrity, authentication, key management, key exchange, and Public-Key Infrastructure (PKI). To achieve this goal, students will implement a simple SSL VPN for Ubuntu.
- *Role-Based Access Control (RBAC) Lab:* In this lab, student will implement an integrated access-control system that uses both capability and RBAC access control mechanisms. The lab tasks require students to modify the Minix kernel to implement the proposed system.
- *Capability lab:* In this lab, students will design and implement a capability-based access control system in Minix. Similar to the RBAC lab, the lab tasks will involve modifying the Minix kernel. The capability-based access control system would co-exist with the file-based permission used in Minix.
- *Encrypted file system lab:* In this lab, students will design and implement an encrypted file system for Minix. A key requirement of the design is to ensure that EFS is transparent to the applications. From the users perspective, there should be no more work other than mounting the new filesystem to be able to use the new file system.
- *Set-random UID lab:* In this lab, students will implement a simple sandbox for executing untrusted programs. In the simple sandbox, programs are assigned randomly generated uid that do not exist in the system. Because the random uid do not own files, the access permissions

of the executed program are limited.

- *Address space randomization lab:* In this lab, students will implement methods for randomizing the heap and stack of the `Minix` operating system.

III. Coverage of SEED Labs

A. Principle Coverage

Regardless of how instructors teach computer security and in what contexts (e.g. networking, operating system, etc.), one thing is for sure: they should cover the principles of computer security. In civil engineering, when building bridges, there are well-established principles that need to be followed. Security engineering is no different: in order to build a software system that is intended to be secure, we also need to follow principles. Regardless of how computer security is taught, the fundamental principles that most instructors cover are quite the same, even though the principles might be covered in different contexts.

The definition of "security principles" is interpreted differently by different people: some interpret it as software engineering principles, such as the principle of least privileges; some interpret it as access control, authentication, etc. To avoid confusion, we use the following definition:

> *A computer security principle is an accepted or professed rule of action or conduct in building a software or hardware system that is intended to be secure.*

We broadly categorized the fundamental computer security principles into the following classes: Authentication (AU), Access Control (AC), Cryptography (CG), Secure Coding (SC), and Secure Design (SD). Table I also shows the security principles covered by each of the SEED labs.

B. Courses Coverage

After studying a number of security courses taught at different universities and colleges, we identified several representative types of courses, and provide suggestions regarding what SEED labs are appropriate for these courses (Table I).

System-focused Courses: This type of course focuses on security principles and techniques in building a software system. Network, also considered as a system, might be part of the course, but not the main focus. The focus is mainly on software systems in general. Operating systems, programs, and web applications are usually used as examples in these courses.

If an instructor wants to ask students to design and implement a real system related to system security, there are several choices. (a) If the instructor wants to let students learn how to use cryptography in a real system, the Encrypted File System Lab is a good choice. (2) If the instructor wants to let students gain more insights on access control mechanisms, the Role-Based Access Control Lab and Capability Lab are good candidates. (3) If the instructor wants students to learn some of the interesting ideas in improving system security, the Address Space Layout Randomization Lab and the Set-RandomUID Sandbox Lab are good candidates. Because all these labs require modifying the underlying operating system kernel, these are labs are meant to be carried out in the `Minix` operating system. These labs can be used as the final projects.

Networking-focused Courses:: This type of a course focuses mainly on the security principles and techniques in networking.

Programming-focused Courses:: The goal of this type of course is to teach students the secure programming principles when implementing a software system. Most instructors will cover a variety of software vulnerabilities in the course.

Software-Engineering-focused Courses:: This type of a course focuses on the software engineering principles for building secure software systems. For this type of courses, all the vulnerabilities labs can be used to demonstrate how flaws in the design and implementation can lead to security breaches. Moreover, to give students an opportunity to apply the software engineering principles that they have learned from the class, it is better to ask students to build a reasonably sophisticated system, from designing, implementation, to testing. Our design/implementation labs can be used for this purpose.

C. Textbook Coverage

Most of instructors do choose a particular textbook in their courses. There are several popular textbooks that are popular among the computer security instructors. We show how SEED labs can be used along with those textbooks; in particular, for each of the textbooks, we have identified the chapters that can use our SEED labs as their lab exercises to enhance student's learning of the subjects in those specific chapters. Our results are summarized in Table II. We have picked the following four textbooks in our studies:

- *Introduction to Computer Security*, by Matt Bishop (published by Addison-Wesley Professional in October 2004). We refer to this book as *Bishop I*.
- *Computer Security: Art and Science*, by Matt Bishop (published by Addison-Wesley Professional in December 2002). We refer to this book as *Bishop II*.
- *Security in Computing (3rd Edition)*, by Charles P. Pfleeger and Shari Lawrence Pfleeger (published by Prentice Hall PTR in 2003). We refer this book as *Pfleeger*.
- *Network Security: Private Communication in a Public World (2nd Edition)*, by Charlie Kaufman, Radia Perlman, and Mike Speciner (published by Prentice Hall PTR in 2002). We refer this book as *KPS*.

IV. Evaluation

With the goal of this project being enhancement of student learning in security education via laboratory exercises, the focus of the current assessment effort is on the efficiency and effectiveness of the SEED labs. The efficiency of the lab exercises is quantified by the following metrics: level of difficulty, time spent on lab is worthwhile, and clarity of instructions. The effectiveness of the lab exercises assesses student learning and is quantified by the following metrics: student level of interest, attainment of learning objective, and value of the lab as part of the curriculum. Students were asked

Labs	Bishop I	Bishop II	Pfleeger	KPS
Buffer-Overflow Lab	20, 26	23, 29	3	-
Return-to-libc Lab	20, 26	23, 29	3	-
Race-Condition Lab	20, 26	23, 29	3	-
Format-String Lab	20, 26	23, 29	3	-
Chroot Sandbox Lab	20, 26	23, 29	3	-
TCP/IP Attack Lab	20, 23, 26	23, 26, 29	3	-
DNS Pharming Attack Lab	20, 23, 26	23, 26, 29	3	-
Cross-Site Scripting Attack Lab	20, 23, 26	23, 26, 29	3	25
Cross-Site Request Forgery Attack Lab	20, 23, 26	23, 26, 29	3	25
ClickJacking Attack Lab	20, 23, 26	23, 26, 29	3	25
SQL Injection Attack Lab	20, 23, 26	23, 26, 29	3, 6	-
Set-UID Program Vulnerability Lab	14	15	4	-
Pluggable Authentication Modules Lab	11	12	4.5	9, 10
Linux Capability Exploration Lab	12, 14, 17	13,15,19	4	-
Secret-key Encryption Lab	8-10	9,10,11	2,12	2-6
One-Way Hash Function Lab	8-10	9,10,11	2,12	2-6
Public-key Crytography Lab	8-10	9,10,11	2,12	2-6
SYN-Cookie Lab	23	26	2, 7	5
Packet Sniffing and Spoofing Lab	23	26	2, 7	5
Web Access Control Lab	4, 14	4, 15	4, 7	25
Set-RandomUID Sandbox Lab	19.6	22.7	-	-
Minix Capability Lab	12, 14, 17	13,15, 19	4	-
Minix Role-Based Access Control Lab	12, 14, 17	13, 15, 19	4	-
Encrypted File System Lab	8-10, 17	9-11, 13, 19	2, 4	2-5
Address-space Layout Randomization Lab	22, 24, 26	25, 27, 29	4, 5	-
IP Sec Lab	8-10, 17, 23	9-11, 19, 26	2, 7	2-5, 17
VPN Lab	8-10, 17, 23	9-11, 19, 26	2, 7	2-5, 17
Linux Firewall Lab	17, 23	19, 26	7.4	23
Minix Firewall Lab	17, 23	19, 26	7.4	23

TABLE II
TEXTBOOK MAPPINGS (THE NUMBERS IN THE TABLE ARE CHAPTER NUMBERS)

to submit a short questionnaire after completing each lab based on these metrics. Student responses were confidential and completion of the surveys was on a voluntary basis. Among all the SEED labs we have developed (about 25), 11 of them have statistically sufficient number of survey responses (we have total 735 survey responses). Therefore, The survey results only from 11 individual laboratory exercises were analyzed.

Eight of the 11 exercises are categorized as Vulnerability and Attack Labs, with their overall learning objective focused on students learning the principles of secure design, programming, and testing. Three of the 11 exercises are identified as Design/Implementation Labs, with their overall learning objective focused on students applying security principles in designing and implementing systems. Survey data from 5 of the 11 labs were gathered over 3 years of implementation (2007-2009), with Total N for these labs ranging from 39-77 student respondents. Four of the 11 labs were included in the curriculum for 2 of the 3 years, with 3 of these labs not included in 2009. Total N for these labs ranged from 37-60 student respondents. During 2009, 2 new labs were introduced, with Total N ranging from 11-15 student respondents. Student survey data analysis across all 11 laboratory exercises is summarized below.

- Lab instructions were clear: Approximately, three-quarters or more of respondents (range from 93% to 73%) agreed or strongly agreed for 10 of the 11 labs. 60% of respondents indicated similarly for 1 lab.
- The time I spent on the lab was worthwhile: Approximately, three-quarters or more of respondents (range from 94% to 74%) agreed or strongly agreed for all 11 labs.
- The lab was a valuable part of this course: Over 90% of respondents (range from 99% to 91%) agreed or strongly agreed for 10 of the 11 labs. 82% of respondents indicated similarly for 1 lab.
- I have attained the learning objective of the lab: Over three-quarters of respondents (range from 95% to 78%) agreed or strongly agreed for all 11 labs.
- Student level of interest in the lab: Approximately, three-quarters or more of respondents (range from 92% to 74%) reported a high or very high interest level for 10 of the 11 labs. 66% of respondents indicated similarly for 1 lab.
- Level of difficulty of the lab: Over one-half of respondents (range from 100% to 54%) reported a somewhat difficult or very difficult level for 8 of the 11 labs. 46% - 32% of respondents indicated similarly for 3 labs.

To further explore these results, survey responses for these items were analyzed by student gender, year of lab administration, level of student preparation, student familiarity with

Unix, and level of lab difficulty. Due to page limitation, we are unable to put all the evaluation results in this paper. Detailed evaluation and diagrams can be found from our project web site.

V. Conclusion

To help enhance student's learning in computer security education, we have developed a suite of hands-on lab exercises. These labs can be conducted in an environment that is very easy to build using the computing facilities that are already available to students in most universities. Based on this environment, instructors can select among 25 well-developed laboratory exercises for their courses. These labs cover a wide spectrum of security principles and can accommodate a variety of security courses. We have experimented with these labs in our own courses for the last seven years. The results are quite promising. More than 15 other universities have also used our labs, and their feedbacks are also quite positive.

We would like to disseminate the SEED labs to a larger audience of instructors. All our labs are publicly available for use under the open-source license. Instructors are welcome to modify our lab description, if they want to tailor the lab descriptions to fit their courses. The entire lab environment is built into a pre-built virtual machine image, and can be downloaded from our web page. We have dedicated set of assistants, funded from the grant provided by NSF, for providing help to instructors if they run into problems in using our lab environment.

Acknowledgment

We thank the anonymous reviewers for their comments and kind suggestions. This work was partially supported by Award No. 0618680 from the United States National Science Foundation.

References

[1] P. J. Denning. Great principles of computing. *Communications of hte ACM*, 46(11):15–20, November 2003.

[2] J. M. D. Hill, C. A. C. Jr., J. W. Humphries, and U. W. Pooch. Using an isolated network laboratory to teach advanced networks and security. In *Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education*, pages 36–40, Charlotte, NC, USA, February 2001.

[3] C. E. Irvine. Amplifying security education in the laboratory. In *Proceeding IFIP TC11 WC11. First World Conference on INFOSEC Education*, pages 139–146, Kista, Sweden, June 1999.

[4] C. E. Irvine, T. E. Levin, T. D. Nguyen, and G. W. Dinolt. The trusted computing exemplar project. In *Proceedings of the 2004 IEEE Systems Man and Cybernetics Information Assurance Workshop*, pages 109–115, West Point, NY, June 2004.

[5] D. Kolb. *Experiential learning: Experience as the source of learning and development*. Prentice Hall, Englewood Cliffs, NJ, 1984.

[6] W. G. Mitchener and A. Vahdat. A chat room assignment for teaching network security. In *Proceedings of the 32nd SIGCSE technical symposium on Computer Science Education*, pages 31–35, Charlotte, North Carolina, United States, 2001. ACM Press.

VI. Author Biographies

Wenliang Du is an associate professor of computer science in the department of EECS, Syracuse University. He obtained his PhD in computer science from Purdue University. Dr. Du's research interests include security education, web security, privacy-preserving data mining, and security in wireless sensor networks.

Karthick Jayaraman is a PhD candidate at the department of EECS, Syracuse University. His advisors are Dr. Wenliang Du and Dr. Steve J. Chapin. His research interests are security education, system, and web security.

Noreen Gaubatz is the Assistant Director of the Office of Institutional Research and Assessment at Syracuse University. She obtained her Ph.D. in Higher Education Administration from Syracuse University. Dr. Gaubatzs work supports a variety of university-wide assessment initiatives, with her research interest in student ratings of teaching effectiveness.