Laboratory Experiments for Network Security Instruction

JOSÉ CARLOS BRUSTOLONI

University of Pittsburgh

We describe a sequence of five experiments on network security that cast students successively in the roles of computer user, programmer, and system administrator. Unlike experiments described in several previous papers, these experiments avoid placing students in the role of attacker. Each experiment starts with an in-class demonstration of an attack by the instructor. Students then learn how to use open-source defense tools appropriate for the role they are playing and the attack at hand. Threats covered include eavesdropping, dictionary, man-in-the-middle, port-scanning, and fingerprinting attacks. Defense skills gained by students include how to forward ports with OpenSSH, how to prevent weak passwords with CrackLib, how to salt passwords, how to set up a simple certifying authority, issue and verify certificates, and guarantee communication confidentiality and integrity using OpenSSL, and how to set up firewalls and IPsec-based virtual private networks. At two separate offerings, tests taken before and after each experiment showed that the experiments have statistically significant and large effect on students' learning. Moreover, surveys show that students finish the sequence of experiments with high interest in further studies and work in the area of security. These results suggest that the experiments are well-suited for introductory security or networking courses.

Author's address: J. Brustoloni, Department of Computer Science, University of Pittsburgh, 210 S. Bouquet St. #6111, Pittsburgh, PA 15260, USA, email: jcb@cs.pitt.edu, Web: http://www.cs.pitt.edu/~jcb/.

This project was funded in part by an Innovation in Education Award from the University of Pittsburgh's Advisory Council on Instructional Excellence and in part by The Technology Collaborative through a grant from the Commonwealth of Pennsylvania, Department of Community and Economic Development.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; D.4.6 [**Operating Systems**]: Security and Protection—*Authentication*; *Cryptographic controls*; E.3 [**Data**]: Data Encryption—*Public key cryptosystems*; *Standards*; H.1.2 [**Models and Principles**]: User/Machine Systems—*Human factors*

General Terms: Security, Experimentation
Additional Key Words and Phrases: Certificate, certifying authority, course, dictionary attack, eavesdropping, education, experiment, fingerprinting, firewall, IPsec, man-in-the-middle, password, port scanning, security, SSH, SSL, VPN

1. INTRODUCTION

The accelerating number and sophistication of attacks against computer networks and systems has markedly increased the demand for computer security professionals. This demand has recently motivated a significant increase in the number of security courses in computer science curricula [Vaughn Jr. 2000; Logan and Clarkson 2005]. Hands-on activities are the centerpiece in many of these new courses. In many cases, these activities include "cyberwar" exercises where students alternately learn how to attack and defend computer systems [Hill et al. 2001; Micco and Rossman 2002; Mateti 2003; Wagner and Wudi 2004].

The decision to teach students how to attack computer systems is not without controversy, however [Frincke 2003; Aycock and Barker 2005]. Critics like [Logan and Clarkson 2005] point out the absence of hard evidence that such activities promote effective learning and the risk that some students may misuse what they learn. Several ways to mitigate risks have been proposed, including screening students, teaching the applicable law and ethics and requiring students to sign commitments to appropriate behavior, using isolated networks for attack experimentation, monitoring students, and punishing those that misbehave [Ragsdale et al. 2003; Aycock and Barker 2005]. These safeguards can be difficult to implement in introductory security courses and in continuing education. Their requirement may also be problematic for computer science departments whose staffing and enrollment or budget and space do not permit the creation of stand-alone security courses [Mullins et al. 2002] or isolated testbeds [Bhagyavati et al. 2005].

This paper describes a sequence of five experiments on network security. In these experiments, the instructor demonstrates to students a variety of threats, including eavesdropping, dictionary, man-in-the-middle (MITM), port-scanning, and fingerprinting attacks, and then walks students through the process of defending computer systems and networks against them. The experiments require only an inexpensive and remotely accessible testbed, and avoid placing students in the attacker's role, which remains with the instructor. Because this "instructor plays the attacker" model does not teach students active attacks, it may be safer than the "cyberwar" model, yet still manages to pique students' interest and motivate students to learn how to implement appropriate defenses.

The sequence of experiments is organized top-down, in a manner analogous to the popular networking textbook, "Computer Networks: A Top-Down Approach Featuring the Internet" [Kurose and Ross 2004]. The experiments are suitable for introductory security or networking courses and continuing education. The experiments place students in a variety of roles and teach students how to use open-source defense tools, protocols, and algorithms suitable for each role. Ability to program in C is a pre-requisite. The first experiment places students in the role of computer user, using OpenSSH [Lonvick 2004; Barrett and Silverman 2001] at the application layer. The next three experiments place the user in the role of programmer, using OpenSSL [Freier et al. 1996; Dierks and Allen 1999; Viega et al. 2002] at the application and transport layers. The last experiment casts students as system administrators, using IPsec [Kent and Atkinson 1998] and firewalls [Cheswick et al. 2003] at the network layer.

We tested the experiments by giving students tests before and after they did each experiment. The tests show that the experiments have a statistically significant and large effect on students' learning. End-of-course surveys indicate that students complete the experiment sequence with high confidence and interest in further studies and work in the area of security.

The rest of this paper is organized as follows. Section 2 describes the laboratory setup needed for the experiments. Section 3 describes the first experiment, where students learn how to use OpenSSH and port forwarding. Section 4 describes the

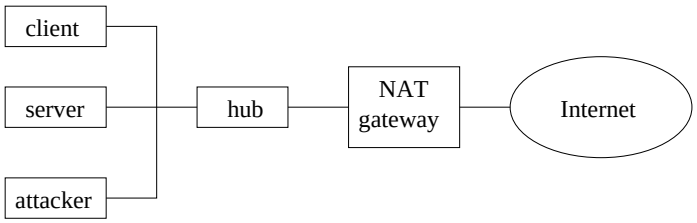

Fig. 1. Cluster configuration for the first four experiments. The hub facilitates packet sniffing by the attacker. The NAT gateway prevents packet sniffing outside the cluster. The NAT gateway is configured to forward SSH to the client node, so that remote users can securely access cluster computers.

second experiment, which teaches students how to proactively filter weak passwords, salt passwords, and limit authentication attempts. The third experiment, described in Section 5, teaches students how to use the OpenSSL library for communication confidentiality and integrity. Section 6 describes the fourth experiment, where students learn how to use OpenSSL to set up a simple certifying authority and how to issue and verify certificates so as to avoid MITM attacks. The fifth experiment, described in Section 7, teaches students how to set up a firewall and a virtual private network (VPN). Section 8 presents results of assessments of the experiments in two separate offerings, and Section 9 discusses these results. Section 10 discusses related work, and Section 11 concludes.

2. LABORATORY SETUP

This section describes the laboratory necessary for the experiments.

Security experiments often need to violate university policies or would be unacceptably disruptive on general-purpose networks. We therefore set up a new laboratory specifically for security instruction. The new laboratory uses private networks [Rekhter et al. 1996] that can be reconfigured to emulate a variety of attack scenarios. Because the attacks are confined to the experimental testbed, they do not disturb the campus network.

For the experiments described in this paper, the laboratory is organized into *clusters* each comprising three personal computers connected via a Fast Ethernet hub. In the first four experiments, the computers run, respectively, a *client*, a

server, and an *attacker* application, as illustrated in Fig. 1. Unlike switches, hubs use a shared medium for communication. Use of a shared medium makes it simpler for the attacker node to sniff packets sent between client and server, for attack or debugging purposes. The attacker would still be able to sniff packets if a switch were used, but would need to use additional tools [Song 2000].

We found it advantageous to make the laboratory's clusters remotely accessible, to the extent allowed by the experiments being performed. (When a cluster is used for an experiment that is potentially too disruptive, such as denial of service, the cluster needs to be isolated and cannot be remotely accessible. However, the five experiments described in this paper do not have that characteristic.) Remote access enables the instructor to demonstrate attacks to students in class, in preparation for each new experiment. (We usually connect a classroom computer's video output to an LCD projector, and open one window for each remote computer.) Remote access also allows students to implement and test all or part of their solutions according to their own schedule.

We enabled remote access by interposing a network address translator (NAT) gateway between each cluster and the Internet [Srisuresh and Holdrege 1999]. NAT is necessary because the private addresses used within each cluster are not unique and therefore not routable on the Internet. We configured each cluster computer to start an SSH daemon at boot time, and set up NAT's port forwarding to forward SSH packets to the client node. This configuration enables remote users to initiate an SSH session with the client node and open a secure shell or transfer files securely to or from the client node (e.g., using scp or sftp). Using a secure shell on the client node, a user can then (1) employ SSH to open a secure shell on the server or attacker nodes, or (2) use scp or sftp for securely transferring files between client and server or attacker nodes. Each user can have any number of SSH sessions in each cluster computer.

The laboratory currently has four clusters. At our past enrollment levels, each cluster was shared by up to eight students and seemed to be able to handle higher loads. Each cluster can be set up very inexpensively. Most routers for home networks include a suitable NAT gateway. Current online prices for the 4-port Fast

Ethernet hub and Wi-Fi router/NAT gateway we used are $38 and $120, respectively (plain Ethernet hubs and home routers without Wi-Fi are even less costly). Each cluster requires three computers, but these can be older computers that would otherwise be retired (the current online price of new computers that would be more than adequate is about $600 each, including LCD monitor). The laboratory's computers currently run FreeBSD 4.8 [FreeBSD 2003]. Linux or other operating systems would also be possible. We configured the computers with static addresses, and disabled DHCP (Dynamic Host Configuration Protocol) in the clusters. DHCP is commonly used in home routers, but it may assign variable addresses to each computer. Variable addresses can be confusing, especially for users who are accessing clusters remotely.

Each student receives an unprivileged account with machine-generated password in the computers of one of the clusters. Student usernames and passwords are entered into a table on removable media (e.g., floppy disk). Automated scripts use this table for creating or deleting students' accounts in each cluster computer. A student's machine-generated password is the same in each computer of a cluster, but password databases are local to each computer (this enables the user to log into a computer locally even if the computer's network configuration is disrupted). A student may manually change her password in each computer where she has an account. However, the instructor advises students not to use in the laboratory passwords that they use elsewhere. Unprivileged accounts are less risky than administrative ones because unprivileged users cannot change a computer's configuration. However, not all experiments can be done remotely or with an unprivileged account. In particular, only the first part of the fifth experiment can be done in this manner. To complete that experiment, students need to schedule a laboratory slot to use a cluster exclusively, with help from the instructor or a teaching assistant. The instructor and teaching assistants have administrative accounts. During each slot, one student in each of the clusters can complete his or her experiment (e.g., four students per two-hour slot, if there are four clusters).

The attacker node is configured with `/dev/bpf*` (Berkeley packet filter) readable not only by its owner (root), but also by others. This configuration allows unpriv-

leged users (e.g., students) to use packet sniffing applications (e.g., tcpdump [Tcpdump 2003] or ethereal [Ethereal 2003]) for debugging purposes. The NAT gateway prevents students from sniffing outside the cluster, because cluster computers are in a private subnetwork. In all other respects, the configuration of cluster computers normally conforms to university policies (in particular, students have only unprivileged accounts). Students change the configuration of cluster computers only in the fifth experiment, and are monitored when they do so.

We gave students two to three weeks to perform each experiment. We distributed instructions for each experiment right after the previous experiment's solution was due. Since experiments 2 through 4 build on the respective previous experiment's solution, we distributed sample solutions for experiments 1 through 3 several days after they were due. Late submissions for those experiments cannot be accepted after the respective sample solution is distributed.

3. EAVESDROPPING ATTACKS / SSH

This section describes our first experiment.

This experiment's goals are (1) to teach or remind students how to design and implement a simple client/server application using sockets and TCP/IP, (2) to make students aware of the insecurity of default passwords, printed passwords, and passwords transmitted in plaintext, and (3) to teach students how to eliminate these vulnerabilities by modifying the application or using the SSH secure shell.

The instructor shows in class how to use SSH for accessing the laboratory remotely, how to open windows for different laboratory computers, and how to transfer files to and from the laboratory or between laboratory computers. The instructor demonstrates a simple client/server application, where the server authenticates the client by checking a password and then accepts client commands for retrieving files. The instructor shows how an attacker can use tcpdump or other sniffing program to monitor the packets sent between client and server. In particular, the instructor highlights how easily visible the client's password is. The instructor then shows how the same application can be used, without modification, with SSH port forwarding [Barrett and Silverman 2001], and demonstrates that attackers can then

no longer see the client's password.

The instructor discusses some requirements for SSH port forwarding to be possible, namely (1) a server application that displays or uses known server port numbers, (2) a client application that allows configuring server addresses and port numbers, (3) an SSH server that accepts port forwarding and runs in the same computer or domain as the server application, and (4) absence of firewall rules for dropping SSH traffic between client and SSH server.

In the first part of the experiment, students are to modify a sample client and a sample server application. The server given to students simply accepts a connection and sends a message to the client. The given client simply connects to the server, receives the server's message, and writes the latter to standard output. Students need to modify the server so that it authenticates clients using passwords and accepts and processes client commands for file retrieval. Students need to make corresponding changes to the client. Sample data and password files are also given to students.

In the second part of the experiment, students are to modify (1) the server application, so that it rejects password files containing default or empty passwords, and (2) the client application, so that it does not print the password on the screen while the user is entering it. The installation of operating systems, computer devices (e.g., Wi-Fi routers), and many server applications, often creates at least one account with administrative rights. Frequently, the installation procedure allows users to accept a default password (or even not select any password at all) for such an account. Attackers can then use the default (or no) password to gain access to the administrative account and install Trojan horses or other malware in the system. The experiment teaches students how to thwart such attacks: a system can be designed such that it forces installers to protect with unique passwords accounts created by the system's installation. Plaintext printing of passwords is another common vulnerability that unnecessarily exposes passwords to potential attackers. The experiment teaches students how to avoid such disclosure.

The third part of the experiment is similar to the initial in-class demonstration, but is now performed in the laboratory, by students, with their own client and

server. Using tcpdump on the attacker node, students show the instructor how eavesdroppers can easily sniff traffic between client and server and obtain client passwords. Students then show how the same client and server can communicate using SSH port forwarding, and demonstrate that passwords then are not exposed to eavesdroppers.

4. DICTIONARY ATTACKS

This section describes our second experiment.

This experiment's goals are (1) to make students aware of dictionary attacks, and (2) to teach students how to defend systems against such attacks by proactively filtering weak passwords, salting passwords, and limiting authentication attempts. In the process of doing the experiment, students also learn about secure hash functions and their implementation in OpenSSL. The latter is a free, open-source implementation of the SSL protocol and includes the implementation of many cryptographic algorithms [OpenSSL 2003; Viega et al. 2002].

The instructor demonstrates in class an attack application that gains access to the previous experiment's server files by guessing passwords from a dictionary. The instructor invites students to suggest alternative passwords, and shows that the attack application is often able to guess them. (Because the demonstration uses an attacker and a server that are on the same private Fast Ethernet, and the server replies as fast as it can, the dictionary attack usually either succeeds almost instantly, or fails. If the password is in the dictionary, the attack invariably succeeds very fast.)

In the experiment, students are to design and implement a program for adding passwords to the server's password file. The program should reject weak passwords. Weak passwords are those that can be found in a dictionary, or simple variations thereof (e.g. those that capitalize one or more letters of a dictionary word, substitute a letter by a number or symbol that looks similar, append a number or symbol, or write the word backwards). Students learn that programs that automate the process of guessing passwords using a dictionary are easily available on the Web (e.g., Crack [Muffet 2003a] and John the Ripper [Openwall 2003]). To mitigate

this problem, students are to use the CrackLib library [Muffet 2003b]. This library tests for hundreds of patterns that users often follow in selecting passwords, and that make passwords predictable by an attacker. (Note that students use CrackLib, the defense tool, not Crack, the attack tool.) Students (and the initial in-class demonstration) use the default dictionary included in CrackLib's distribution.

In addition to filtering weak passwords, the password program should not store plaintext passwords. Instead, each entry in the password file should contain the user id, *salt* (i.e., a random number, in plaintext), and a secure hash of the user id, salt, and password. Secure hash functions, such as SHA [National Institute of Standards and Technology 1995a], can be easily computed by a server application and compared with the value on the password file for client authentication. On the other hand, access to the password file by an attacker does not easily reveal the original passwords, since secure hashes cannot be inverted. An attacker might precompute the hash of all words in a dictionary and use the result to identify weak passwords in the password file. However, if the salt is sufficiently long, such an attack can be impractical.

Students need to modify the server application from the previous experiment, so that it uses the new password file format. Additionally, students need to modify the server so that the latter locks out a client's IP address for a configurable period (e.g., 5 minutes) if a client that uses that IP address has failed authentication more than a configurable number of times (e.g., 3). Lock-out can dramatically reduce the rate at which an attacker who does not have the password file can test password guesses.

5. USING SSL TO ENSURE COMMUNICATION CONFIDENTIALITY AND INTEGRITY

This section describes our third experiment.

This experiment's goal is to teach students how to use OpenSSL to generate private keys, public-key certificates, and ensure confidentiality and integrity of communication between a client and a server. In the process of doing this experiment, students also learn about encryption and the differences between symmetric and

public-key cryptographic algorithms (e.g., AES [National Institute of Standards and Technology 2001b] and RSA [Rivest et al. 1978]) and applications (e.g., bulk encryption and digital signatures).

The instructor tells students that SSH is a good tool for security-conscious *users*, but SSL [Freier et al. 1996] (or its standard equivalent, TLS [Dierks and Allen 1999]) may be a better tool for security-conscious *programmers*. SSH provides secure alternatives to telnet and ftp; SSH's port forwarding can also secure other protocols that send passwords in plaintext, such as pop3. However, a user needs to *remember* to use SSH. On the contrary, if an application's programmer uses SSL, he or she can embed security, such that the application is secure regardless of whether its user is security-conscious.

The instructor gives students a sample client and a sample server OpenSSL application. The client application simply opens an SSL connection to the server and receives and displays a "Hello world!" message from the latter. Students can use the sample client and server as examples for how to use OpenSSL in a program.

Students are to modify the previous experiment's client and server applications so that they communicate securely using SSL, instead of TCP. They should ensure that SSL uses SHA for message integrity and AES for message confidentiality (the sample applications they received use instead MD5 and 3DES). Students also need to use OpenSSL applications to generate the server's private key and corresponding self-signed public-key certificate. They should verify with tcpdump that, after the modification, communication between client and server is encrypted.

The amount of programming necessary for this experiment is less than for the previous two experiments. However, this is compensated by the fact that students need to consult OpenSSL documentation, with which they usually are unfamiliar, to complete the experiment.

6. MAN-IN-THE-MIDDLE ATTACKS

This section describes our fourth experiment.

This experiment's goals are (1) to make students aware of man-in-the-middle (MITM) attacks, (2) to teach students how to use OpenSSL to set up a simple

certifying authority, and (3) to teach students how to thwart MITM attacks by properly verifying certificates.

The instructor demonstrates in class how an attacker can impersonate a server and fool the previous experiment's client to connect to the attacker, instead of to the server. The attacker can then capture the client's password and relay packets between client and server. Although client and server use SSL, the attacker can read, modify, or forge any packets. Two tools are easily available for such attacks, arpspoof and dnsspoof [Song 2000]. Arpspoof's principle of operation is ARP (Address Resolution Protocol) "cache poisoning." The attacker sends the victim ARP replies that wrongly associate the IP address of the victim's default gateway with the attacker's MAC address. This causes the victim to send to the attacker packets destined to the Internet. The attacker also sends the gateway ARP replies that wrongly associate the victim's IP address with the attacker's MAC address. This causes the gateway to send to the attacker packets returning to the victim from the Internet. Dnsspoof's principle of operation is similar. The attacker sends the victim DNS (Domain Name System) replies that wrongly associate a server's hostname with the attacker's IP address. This causes the victim to connect to the attacker, instead of to the server.

The instructor explains in more detail what a certificate is and how a certificate is supposed to be issued by a trusted certifying authority. The instructor then explains how programmers should verify the certifying authority's signature on a certificate and the match between a certificate's subject and the server the client wants to access. Such verification thwarts MITM attacks like the one just demonstrated.

The instructor gives students a shell script that implements, using OpenSSL, the key functions of a certifying authority. These functions include verifying that certificate requests conform to the certifying authority's policies, and signing, verifying, and keeping a record of issued certificates. (The script follows the detailed instructions for setting up a certifying authority using OpenSSL, found in Section 3.3 of [Viega et al. 2002].) Students are to modify the certifying authority's policies, e.g. validity period of issued certificates, and issue the certifying authority's root certificate.

Students then create the server's private key and certificate signing request. They send this request to the certifying authority, which issues the corresponding certificate. Students send the server certificate to the server, and the root certificate to the client.

Students need to modify the previous experiment's server application so that it properly loads the server's certificate and private key. Students also need to modify the client application so that it validates the certifying authority's signature on the server's certificate, and verifies that the subject's common name on the server's certificate matches the fully qualified domain name (FQDN) of the server the client wishes to connect to.

Students should verify that the client application connects with the server if the latter uses a certificate issued by a certifying authority that the client is configured to trust, but not if the server uses the previous experiment's certificate, which was self-signed by the server. Students should also verify that the client application does not connect with a server if the server uses another server's certificate, even if the certificate is issued by a certifying authority that the client is configured to trust (the certificate's common name must match the FQDN of the server the client wishes to connect to).

As extra credit, students are asked to modify the client application so that it also checks the certifying authority's revocation list.

7. FIREWALLS AND VIRTUAL PRIVATE NETWORKS

This section describes our fifth experiment.

This experiment's goals are (1) to teach or remind students how to configure network interfaces and routing tables, (2) make students aware of port scanning and fingerprinting attacks and their use for remote takeover, and (3) teach students how to thwart such attacks by using firewalls [Cheswick et al. 2003] and virtual private networks [Kent and Atkinson 1998].

The instructor explains to students that system administrators cannot guarantee network security simply by expecting users to use SSH or programmers to use SSL. The instructor then explains the concept of a security perimeter and how

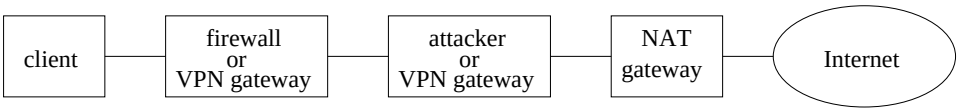

Fig. 2. Cluster configuration for the fifth experiment. Configuration of the firewall prevents the attacker from port-scanning or fingerprinting the client. Configuration of the IPsec-based VPN gateways guarantees confidentiality and integrity of the packets tunneled between the gateways, independently of user commands (e.g., SSH) or application modifications (e.g., SSL).

system administrators implement it using firewalls and virtual private networks. The instructor demonstrates a port scanning and fingerprinting attack [Insecure.org 2003] and discusses how such attacks may give attackers information needed for a remote takeover. In our setup, the port scanning and fingerprinting attack takes a couple of minutes. The instructor then demonstrates how a firewall can stop such attacks.

For this experiment, each cluster needs to be configured as shown in Fig. 2. As usual, each computer has an SSH daemon active. Before coming to the laboratory, students create private keys and respective public-key certificates for the certifying authority and VPN gateways. Students also create IPsec configuration files for the VPN gateways.

In the laboratory, each student first configures the respective cluster's cables, network interfaces, and routing tables as shown in Fig. 2. The student demonstrates to the instructor that the client can access a Web site via the Internet, and that the attacker can ping the client.

The instructor then uses nmap to demonstrate to the student that the attacker can figure out what ports are open in the client (SSH, port 22) and what operating system the client is running. The student then configures stateful firewall rules for dropping any packets destined to the client that do not belong to a connection initiated by the client. The instructor uses nmap to verify and demonstrate that the attacker can then no longer find any ports open or fingerprint the client, which is behind the firewall.

The student then deletes the firewall and configures the VPN. The student demonstrates to the instructor that the client can still access a Web site via the

Internet. The student also uses tcpdump in one of the VPN gateways to show that the packets sent between the VPN gateways are tunneled using IPsec.

Most students complete the laboratory part in less than two hours. Note that one of each cluster's computers remains connected to the Internet, allowing other students to prepare their files remotely.

8. ASSESSMENT

This section reports the results of our evaluation of the experiments.

We offered the experiments twice. Our department did not at the time offer a stand-alone course on security. Thus, the first offering was integrated with an undergraduate networking course, and the second offering was integrated with a graduate distributed systems course. No student took both courses. The courses were organized around the textbooks "Computer Networks: A Top-Down Approach Featuring the Internet" [Kurose and Ross 2004] and "Distributed Systems: Principles and Paradigms" [Tanenbaum and van Steen 2002], respectively. These courses were modified only minimally to accommodate the security experiments.

Each course's syllabus contained a statement about ethical use of the laboratory and warning students against attacking other users. Since we did not actually give students any active attack tools (in the experiments, the attacker is always the instructor), ethical issues were discussed only briefly in the first class. Periodically, the instructor made demonstrations in class to introduce each experiment, as described in Sections 3 to 7, taking an average of about 20 minutes for each experiment. There were no other lectures or assignments on security until the fifth experiment, when the chapter on security in the respective textbook was covered. The handouts for each experiment contained a short summary of the main security concepts covered in the experiment (one to two pages), along with Web links and experiment instructions. Presumably, students' learning on security was due mostly to performing the experiments, and not to lectures or the textbook.

We evaluated each experiment by giving students tests before and after they did each experiment. Students were not allowed to keep test questions and were not provided the expected answers. With two exceptions, we used objective tests con-

taining 20 true/false statements on topics covered by the respective experiment. Post-tests covered the same topics as the respective pre-tests, but with different wording. The exceptions were Experiment 4's pre- and post-test in the first offering, which were performance-based [Hart 1992]. In those tests, a teaching assistant observed whether the student performed each of three tasks securely or not (where "securely" meant "avoiding a MITM attack"), before and after doing the experiment.

Results are shown in Tables I and II. Reported p-values are two-tailed. For all experiments other than the fourth one in the first offering, the p-value was determined by paired t-test, and Cohen's effect size, d, was calculated by dividing the mean of the paired differences by the latter's standard deviation (σ). For the fourth experiment in the first offering, we calculated the p-value using Wilcoxon's signed ranks test, since the dependent variable (i.e., test scores) had only a few possible values and therefore could not be considered to be normally distributed. In this case, the reported effect size is:

$$\eta^2 = \frac{\chi^2}{n(p-1)},$$

where n is the number of pairs (12 in this case) and p is the number of tests (2 in this case). By usual convention, effect sizes are considered large if d $\geq$ 0.8 [Cohen 1988] or $\eta^2 \geq 0.14$ [Morse 1999]. P-values less than 0.05 are considered significant [Cohen 1988]. All experiments other than the fourth one consistently had significant and large effect. The fourth experiment had significant and large effect in the first offering, but in the second offering had medium effect. The latter result is significant if it is assumed to test a one-tailed hypothesis (i.e., only positive effects are of interest, in which case the p-value is half of that reported on the table).

We surveyed students' perceptions and attitudes toward security at the end of each course. Each survey question asked whether the student strongly agreed, agreed, disagreed, or strongly disagreed with a statement. Statement 1 was "Laboratory experiments helped me learn how to apply security principles and tools in practice." Statement 2 was "I do not believe I could have learned as much about security simply from lectures." Statement 3 was "Demonstrations in class and in the laboratory made me aware of contemporary security threats and what I need

Exp.	n	Pre-test		Post-test		Paired t-test		Effect size	p-value
		Mean	σ	Mean	σ	Mean	σ		
1	20	15.45	2.11	17.55	2.06	2.10	2.10	1.00	< 0.0005
2	15	13.47	2.48	19.00	1.00	5.53	2.72	1.96	< 0.0005
3	22	14.45	2.43	17.00	2.18	2.55	2.86	0.89	< 0.0005
4	12	0.33	0.65	1.83	1.17	n.a.	n.a.	0.67^*	0.005
5	12	13.58	1.73	15.58	1.98	2.00	1.76	1.14	0.002

Table I. Evaluation of the experiments in an undergraduate networking class. For experiments 1, 2, 3, and 5, the maximum possible score was 20, and the reported effect size is Cohen's d. For experiment 4, the maximum possible score was 3, and (*) the reported effect size is η^2. All experiments had a significant and large effect.

Exp.	n	Pre-test		Post-test		Paired t-test		Effect size	p-value
		Mean	σ	Mean	σ	Mean	σ		
1	18	15.78	1.83	17.72	1.18	1.94	1.83	1.06	< 0.0005
2	19	14.84	2.39	18.26	0.87	3.42	2.32	1.48	< 0.0005
3	18	16.17	1.98	17.94	1.63	1.78	2.18	0.81	0.003
4	14	15.57	1.60	16.50	1.61	0.93	1.90	0.49	0.09
5	17	13.18	2.74	16.53	2.04	3.35	2.60	1.29	< 0.005

Table II. Evaluation of the experiments in a graduate distributed systems course. For all experiments, the maximum possible score was 20, and the reported effect size is Cohen's d. Experiment 4 had medium effect, and all other experiments had large effect.

to do as a user or programmer to counter them." Statement 4 was (for undergraduate and Master's students) "I believe practical experience in network security is a sought-after skill in the job market" or (for Ph.D. students) "I believe practical experience in network security will be useful in my research area." Statement 5 was "I would be interested in a job involving computer network or system security." Statement 6 was "I am interested in pursuing further studies in the area of computer network or system security." Statement 7 was "I liked this course's emphasis

on security." Statement 8 was "I would like the university to offer more courses in the area of security." The number of respondents was 23 for the undergraduate course and 17 for the graduate course. Table III shows the results we obtained.

In the survey, we also asked open-ended questions about what students liked best or least about the course and suggestions for improvement. We summarize here opinions that were shared by more than one student and related to the experiments. Among undergraduates, 14 students (61%) volunteered that the experiments were what they liked best about the course, and another 3 students (13%) said they liked the way experiments were integrated with the lectures and textbook. Two students disliked that the last experiment's instructions left them little opportunity to come up with their own solutions. Suggestions for improvement included assigning more experiments like the last one, where students physically reconfigure the network (3 students), assigning cyberwar experiments where one student team attacks another (2 students), and doing more in-class demonstrations (2 students). Among the graduate students, 6 (35%) volunteered that the experiments were what they liked best about the course, and another 3 students (18%) said they liked the usefulness of the course activities. Suggestions for improvement included assigning more experiments (3 students) and moving the security experiments to a course specifically about security, devoting experiments in the distributed systems course to topics more closely related to the textbook (3 students).

9. DISCUSSION

The previous section's results suggest that our experiments are highly effective for introducing students to network security. Most experiments had a large effect on students' understanding of network security concepts. Additionally, students ended the sequence of experiments with high confidence and interest in further studies or work in the area.

We assessed the experiments in a manner that greatly deemphasizes lectures and textbook instruction, since the latter covered security extensively only at the time of the fifth experiment. It is interesting to note that students were able to learn a great deal simply by doing the experiments. This outcome may be related

Statement	Undergraduate course	Graduate course
1	96	100
2	91	100
3	100	94
4	96	94
5	78	82
6	83	82
7	100	82
8	96	94

Table III. Percentage of students who agreed or strongly agreed with each survey statement at the end of each course. Students gave the experiments very high approval and finished the course highly interested in further studies or work in the area.

to the choice of topics in the experiments. The experiments cover mostly mature topics (e.g., communication confidentiality and integrity) for which well-understood open-source solutions exist (e.g., OpenSSH, OpenSSL, IPsec, and firewalls). These observations suggest that these experiments could also be effective for continuing or distance education or self-learning.

We used different methodologies to assess the fourth experiment in each offering, and the results are not as consistent as for the other experiments. In the first offering, we did performance-based assessment. This form of assessment attempts to determine not simply what students have conceptually grasped, but also how well students have learned to apply the new concepts in concrete situations. Compared to objective tests, performance-based assessment may be more *authentic* and have greater practical relevance [Hart 1992], although it can be more time-consuming. Performance-based assessment in the first offering seems to have captured greater effect than did the objective tests in the second offering. This might be explained by the hands-on nature of the experiment possibly having a more immediately practical than conceptual impact. On the other hand, a lesser conceptual effect of this experiment than the other experiments would not be entirely surprising. The

fourth experiment requires students to understand public-key cryptography. The latter is generally considered one of the most difficult security concepts for users to understand and use [Whitten and Tygar 1999]. It is possible that to improve student comprehension, more readings, detailed discussion in class, or assignments would be necessary.

It would be interesting to repeat the assessment of the experiments with different instructors and students, preferably at different universities. Although the results reported in this paper are statistically significant, a broader study could better control possible biases (e.g., instructor or student ability).

Based on the students' feedback, we decided to continue offering this sequence of experiments in the undergraduate networking course. These experiments cover not only security concepts, but also skills that an introductory networking course needs to teach, e.g. how to use sockets and TCP/IP in client/server applications and configure network interfaces and routers. The main drawback of these experiments for a networking course is that students do not get to explore transport and routing protocols in depth. We partly compensate these shortcomings by using animations of these protocols in class (e.g., [Kurose and Ross 2004; Holliday 2003]) and encouraging students to use the animations on their own. Students' performance in exams suggests that students' comprehension of networking concepts is not being hurt by the experiments' emphasis on network security.

On the other hand, we decided to offer a stand-alone graduate course on security, as suggested by some students. Our experiments did not integrate with the graduate distributed systems course as well as it did with the undergraduate networking class. The new graduate course will offer the experiments described here for those who are new to security, and alternative experiments for more advanced students.

10. RELATED WORK

In our surveys, some students expressed an interest in experiments where students would perform not only defense, but also attack. Skoudis provides many ideas for developing such experiments [Skoudis 2002], and Wagner and Wudi give related management tips [Wagner and Wudi 2004]. Frincke discusses in [Frincke 2003]

the pros and cons of teaching students attack techniques. Attack understanding can give students a more operational, rather than formal, perspective on security, and may inform and motivate the use and creation of better defenses. However, there is also a risk that some students misuse what they learn. Educators at West Point embrace the attack-understanding philosophy wholeheartedly [Ragsdale et al. 2003], but in a context where students are carefully screened, taught the pertinent law, ethics, and possible sanctions for misbehavior, and monitored. West Point uses a fairly large and isolated laboratory for such experimentation. Logan and Clarkson urge similar precautions, but argue that students preparing to be system administrators may be better served by instruction on proper security auditing, forensics, disaster recovery, and business continuity [Logan and Clarkson 2005].

Most previous discussions of attacks in security experiments assume a "cyberwar" model, where students perform both the attack and defense roles. In our experiments, we use an alternative model, "instructor plays the attacker." Most students have previously heard that a variety of attacks exist, but seeing demonstrations of attacks seems to help motivate students to learn defenses against them. We do not provide pointers to attack tools or teach or encourage students to learn how to perform attacks. Instead, we teach and require students to learn how to use defense tools against demonstrated attacks, and discourage students from performing attacks. Our results suggest that our model can achieve several of the goals of the cyberwar model, but at lower risk and in a manner that may be more suitable for novices who have not yet had the appropriate legal and ethical training. In particular, our model seems to succeed in piquing students' interest in security, making students aware of existing threats, and motivating them to implement appropriate defenses. The cyberwar model is perhaps better geared toward more advanced students, who have been screened and received ethical training, and need better understanding of attacks for future endeavors in advanced development or research.

The cluster described in Section 2 and used in all our experiments can be set up very inexpensively. Wulf proposes an even smaller cluster [Wulf 2003], which however cannot be accessed remotely and is too small for experiments that in-

volve routing, firewalls, or VPNs, like our fifth. The distributed testbed described in [Sadasivam et al. 2005] can be used in more advanced experiments, e.g. involving honeypots. It is connected to the Internet via dedicated DSL links, so as not to disrupt the respective campus networks in case of attack. Tikekar and Bacon present a security laboratory that is connected to the Internet via the campus network [Tikekar and Bacon 2003]. Firewalls block traffic between campus and laboratory hosts, making a dedicated Internet link for such a laboratory unnecessary. Although dedicated links or firewalls may protect the campus network, they do not protect other hosts on the Internet. Therefore, such precautions do not alleviate the requirements of ethical training and possibly screening and monitoring of students, if the latter are taught how to perform attacks.

Users often do not understand security features in existing systems and defeat or fail to configure them [Smith 2003]. A plausible way to overcome this problem is to better educate users. In a recent paper [Xia and Brustoloni 2005], we investigated whether students' performance of the fourth experiment reported here positively affects how prudently students use an unmodified browser (and, in particular, avoid MITM attacks). We show in that paper that training gives benefits that are significant and large, but still insufficient, and demonstrate a novel form of just-in-time instruction, *guidance without override*, that is able to achieve both the desired security and usability.

11. CONCLUSIONS

We described five experiments on network security that cast students successively in the roles of computer user, programmer, and system administrator. Unlike experiments described in several previous papers, these experiments avoid placing students in the role of attacker. Each experiment starts with an in-class demonstration of an attack by the instructor. Students then learn how to use open-source defense tools appropriate for the role they are playing and the attack at hand. Threats covered include eavesdropping, dictionary, man-in-the-middle, port-scanning, and fingerprinting attacks. Defense skills gained by students include how to forward ports with OpenSSH, how to prevent weak passwords with CrackLib, how to salt

passwords, how to set up a simple certifying authority, issue and verify certificates, and guarantee communication confidentiality and integrity using OpenSSL, and how to set up firewalls and IPsec-based virtual private networks. At two separate offerings, tests taken before and after each experiment showed that the experiments have statistically significant and large effect on students' learning. Moreover, surveys show that students finish the sequence of experiments with high interest in further studies and work in the area of security. These results suggest that the experiments are well-suited for introductory security and networking courses.

Availability

The instructions, demonstration software, software given to students, and sample solutions for the experiments described in this paper are freely available by request to the author at jcb@cs.pitt.edu for use in for-credit courses.

Acknowledgments

Elaine Rubinstein provided invaluable help in the statistical analysis of the results.

REFERENCES

Aycock, J. and Barker, K. 2005. Viruses 101. In *Proceedings of SIGCSE*. ACM, 152–156.

Barrett, D. and Silverman, R. 2001. *SSH, the Secure Shell: The Definitive Guide*. O'Reilly, Sebastopol, CA.

Bhagyavati, Aguei-Mensah, S., Shumba, R., and Kearse, I. 2005. Teaching hands-on computer and information systems security despite limited resources. In *Proceedings of SIGCSE*. ACM, 325–326.

Cheswick, W., Bellovin, S., and Rubin, A. 2003. *Firewalls and Internet Security: Repelling the Wily Hacker*, 2nd ed. Addison-Wesley.

Cohen, J. 1988. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum, Hillsdale, NJ.

Dierks, T. and Allen, C. 1999. *The TLS Protocol Version 1.0*. IETF, RFC 2246. [Online] ftp://ftp.rfc-editor.org/in-notes/rfc2246.txt.

Ethereal. 2003. Homepage. [Online] http://www.ethereal.com/, last accessed Mar. 2005.

FreeBSD. 2003. Homepage. [Online] http://www.freebsd.org/, last accessed Mar. 2005.

Freier, A., Karlton, P., and Kocher, P. 1996. The SSL protocol version 3.0. [Online] http://wp.netscape.com/eng/ssl3/draft302.txt, last accessed Mar. 2005.

Frincke, D. 2003. Who watches the security educators? *Security & Privacy*, 56–58.

HART, D. 1992. *Authentic Assessment: A Handbook for Educators.* Addison-Wesley.

HILL, J., CARVER, C., HUMPHRIES, J., AND POOCH, U. 2001. Using an isolated network laboratory to teach advanced networks and security. In *Proceedings of SIGCSE.* ACM, 36–40.

HOLLIDAY, M. A. 2003. Animation of computer networking concepts. *ACM Journal of Educational Resources in Computing 3,* 2 (June), Article 2.

INSECURE.ORG. 2003. nmap. [Online] http://www.insecure.org/nmap/, last accessed Mar. 2005.

KENT, S. AND ATKINSON, R. 1998. *Security Architecture for the Internet Protocol.* IETF, RFC 2401. [Online] ftp://ftp.rfc-editor.org/in-notes/pdfrfc/rfc2401.txt.pdf.

KUROSE, J. AND ROSS, K. 2004. *Computer Networks: A Top-Down Approach Featuring the Internet*, 3rd ed. Addison-Wesley.

LEMOS, R. 2003. Security: Open networks pose dilemma. *news.com.* [Online] http://news.com.com/2009-1033-982324.html?tag=rn, last accessed Mar. 2005.

LOGAN, P. AND CLARKSON, A. 2005. Teaching students to hack: Curriculum issues in information security. In *Proceedings of SIGCSE.* ACM, 157– 161.

LONVICK, C. 2004. *SSH Protocol Architecture.* IETF, Internet Draft. [Online] ftp://ftp.rfc-editor.org/in-notes/internet-drafts/draft-ietf-secsh-architecture-17.txt.

MATETI, P. 2003. A laboratory-based course on Internet security. In *Proceedings of SIGCSE.* ACM, 252–256.

MICCO, M. AND ROSSMAN, H. 2002. Building a cyberwar lab: Lessons learned teaching cybersecurity principles to undergraduates. In *Proceedings of SIGCSE.* ACM, 23–27.

MORSE, D. T. 1999. MINSIZE2: A computer program for determining effect size and minimum sample size for statistical significance for univariate, multivariate, and nonparametric tests. *Educational and Psychological Measurement 59,* 3 (June), 518–531.

MUFFET, A. 2003a. Crack version 4.1: A sensible password checker for unix. [Online] http://www.crypticide.com/users/alecm/security/crack-v4.1-whitepaper.ps.gz, last accessed Mar. 2005.

MUFFET, A. 2003b. Cracklib v2.7. [Online] http://www.crypticide.com/users/alecm/security/-cracklib,2.7.tar.gz, last accessed Mar. 2005.

MULLINS, P., WOLFE, J., FRY, M., WYNTERS, E., CALHOUN, W., MONTANTE, R., AND OBLITEY, W. 2002. Panel on integrating security concepts into existing computer courses. In *Proceedings of SIGCSE.* ACM, 365–366.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. 1995a. *Specifications for Secure Hash Standard.* Federal Information Processing Standards Publication 180-1. [Online] http://www.itl.nist.gov/fipspubs/fip180-1.htm.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. 2001b. *Specification for the Advanced*

Encryption Standard (AES). Federal Information Processing Standards Publication 197. [Online] http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

OpenSSL. 2003. Homepage. [Online] http://www.openssl.org/, last accessed Mar. 2005.

Openwall. 2003. John the Ripper password cracker. [Online] http://www.openwall.com/john/, last accessed Mar. 2005.

Ragsdale, D., Welch, D., and Dodge, R. 2003. Information assurance the West Point way. *Security & Privacy*, 64–67.

Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and Lear, E. 1996. *Address Allocation for Private Internets.* IETF, RFC 1918. [Online] ftp://ftp.rfc-editor.org/in-notes/pdfrfc/rfc1918.txt.pdf.

Rivest, R., Shamir, A., and Adleman, L. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21,* 2 (Feb.), 120–126.

Sadasivam, K., Samudrala, B., and Yang, T. A. 2005. Design of network security projects using honeypots. *Journal of Computing Sciences in Colleges 20,* 4 (Apr.), 282–293.

Skoudis, E. 2002. *Counter Hack.* Prentice-Hall, Upper Saddle River, NJ.

Smith, S. 2003. Humans in the loop: Human-computer interaction and security. *Security & Privacy*, 75–79. [Online] http://www.cs.dartmouth.edu/ sws/papers/humans.pdf.

Song, D. 2000. dsniff. [Online] http://naughty.monkey.org/ dugsong/dsniff/, last accessed Mar. 2005.

Srisuresh, P. and Holdrege, M. 1999. *IP Network Address Translator (NAT) Technology and Considerations.* IETF, RFC 2663. [Online] ftp://ftp.rfc-editor.org/in-notes/pdfrfc/rfc2663.txt.pdf.

Tanenbaum, A. and van Steen, M. 2002. *Distributed Systems: Principles and Paradigms.* Prentice-Hall, Upper Saddle River, NJ.

Tcpdump. 2003. Homepage. [Online] http://www.tcpdump.org/, last accessed Mar. 2005.

Tikekar, R. and Bacon, T. 2003. The challenges of designing lab exercises for a curriculum in computer security. *Journal of Computing Sciences in Colleges 18,* 5 (May), 175–183.

Vaughn Jr., R. 2000. Application of security to the computing science curriculum. In *Proceedings of SIGCSE.* ACM, 90–94.

Viega, J., Messier, M., and Chandra, P. 2002. *Network Security with OpenSSL.* O'Reilly, Sebastopol, CA.

Wagner, P. and Wudi, J. 2004. Designing and implementing a cyberwar laboratory exercise for a computer security course. In *Proceedings of SIGCSE.* ACM, 402–406.

Whitten, A. and Tygar, J. D. 1999. Why Johnny can't encrypt: A case study. In *Proceedings of Usenix Security Symposium.* [Online] http://www.usenix.org/publications/library/-proceedings/sec99/full_papers/whitten/whitten.ps.

Wulf, T. 2003. Implementing a minimal lab for an undergraduate network security course. *Journal of Computing Sciences in Colleges 19,* 1 (Oct.), 94–98.

Xia, H. and Brustoloni, J. 2005. Hardening web browsers against man-in-the-middle and eavesdropping attacks. In *Proceedings of WWW 2005.* IW3C2/ACM. [Online] http://www.cs.pitt.edu/~jcb/papers/www2005.pdf.