Teaching Hands-On Network Security: Testbeds and Live Exercises

G. Vigna

Department of Computer Science
University of California, Santa Barbara
E-mail: vigna@cs.ucsb.edu

Abstract

Teaching practical network security requires the use of tools and techniques to support the educational process and to evaluate the students' newly achieved skills. Two fundamental tools that support a hands-on approach to network security are testbed networks and live exercises. Testbed networks provide a safe environment where the students can experiment with the techniques and security tools that they learn about. Live exercises represent a valuable tool to test the students' newly acquired skills and to teach the students the dynamics of network-based attack and defense techniques. However, testbed networks and live exercises are difficult to set up and to manage. For this reason, there are very few courses that use dedicated network testbeds and/or offer live exercise as an integral part of the class work. This paper describes a series of testbed networks and live exercises that have been used in a graduate-level Computer Science course on network security and intrusion detection. Each testbed network is described in detail and its pros and cons discussed. Then, for each live exercise, the setup, execution, and lessons learned are discussed. The intended audience of this paper is represented by instructors – especially in colleges and universities – who want to start using this type of instructional tools but have no experience and are unsure of the possible pitfalls in their design and implementation.

Introduction

Computer security has become a critical issue that affects our everyday life. For this reason, most colleges and other educational institutions have devoted a consistent amount of resources to develop courses and curricula that involve security training (Bishop 1997, Bishop 2000). Typical courses include cryptography, general computer security, network security, and specialized topics, such as firewalls, security of wireless networks, etc.

Most of these courses are taught using standard educational tools, such as textbooks, slides, and papers. In addition, assignment and tests are mostly paper-based and theoretical.

In few instances, courses are characterized by a practical, hands-on approach. These courses are seldom offered because of the additional difficulties of teaching practical security. First of all, security permeates a wide range of technologies. Addressing a comprehensive set of practical security techniques without getting lost in the details of each technology requires careful selection of the topics and particular attention to the way the topics are presented to the students. Second, the instructor is faced with the

difficult decision of choosing the right balance between a completely theoretical approach and a completely practical one (Bishop 1999). Usually, this choice is heavily influenced by the educational curriculum. For example, if there is an existing course that covers the foundational principles of security (e.g., the Bell-LaPadula model), then it is possible to use the course as a prerequisite and focus on more technical issues. Third, teaching practical security requires substantial effort on the part of both the instructor and the educational institution within which the course is given. This makes it difficult to organize practical security courses, particularly in higher education public institutions, like universities, where resources are scarce.

In addition, if the class has a high-impact technical content (e.g., the class covers break-in techniques), then there is a general concern that the class may get 'out of hand' or that a particular institution may be flagged as a 'hacker school' (Brandt 2003).

Nonetheless, we advocate that teaching practical security is important because of the critical nature of the topic. A security education curriculum that covers only the theoretical aspects of security and does not give the student the opportunity to experiment in practice with security technologies cannot prepare the students to the complexity and difficulties of doing research and development in the computer security field.

This paper describes the author's experience in teaching a graduate-level course on 'Network Security and Intrusion Detection'. This course has been taught three times in the past two years. This class differs from traditional courses in security in three ways: a strong practical and technical emphasis, the support for hands-on experience through the use of testbed networks, and the use of live exercises. These three aspects are discussed in the following three sections.

Teaching Practical Security

The goal of the course is to describe in detail the techniques used to violate the security of computer systems and the mechanisms and tools used to both prevent and detect the attacks. The course follows a hands-on approach that gives the students the skills necessary to actually reproduce the attacks and develop the defense tools. For example, the lack of boundary checks in software is not just covered from a theoretical, general viewpoint: Buffer overflow attacks are examined in detail, showing exactly how to create the necessary conditions for an attack to be successful and the tools needed to build and deliver the attack. In addition, the tools used to prevent and detect these attacks are analyzed in details, highlighting the strengths and limitations of each solution.

The rationale behind this approach can be summarized by the well-known saying: 'The Devil is in the details.' That is, only by understanding the low-level details of vulnerabilities and attacks it is possible to avoid the introduction of similar flaws in software and to design protection and detection mechanisms that are actually effective.

A possible argument against this approach is that students are taught 'How to break in'. Some instructors (and some administrators) fear that by teaching how attacks are

actually designed and delivered will create a new generation of hackers that will participate in illicit activities using the skills gained in the security courses.

The obvious counter-argument is that locksmiths know how to break into a house or a car. Locksmiths are not considered criminals and the best locksmiths are those who design the safes where we store our most valuable items. In addition, even though the information about how to violate the security of a system was once in the hands of a few knowledgeable and skilled programmers, nowadays the same knowledge is accessible to the public through the Internet.

Bruce Schneier comments on a recent controversy about teaching viruses are shared by many security experts:

'[…] If we have any hope of improving computer security, we need to teach computer security. Teaching computer security includes teaching how attacks work. It includes teaching how viruses work. It includes teaching how worms work. The bad guys have all sorts of resources to learn how to write viruses. SQL Slammer source code has been available on the Internet. Neither of these two actions will help the bad guys. But they probably will help the good guys. Worms, viruses, exploits, hacking code...they're not infectious diseases. We need to look at them as educational tools, and not things to keep secret.' (Schneier 2003)

The key here is an ethical approach to security. It is the responsibility of the instructor to sensitize the students to the ethical aspects of security and to inform them of the possible consequences of their actions. Therefore, a security class must always include a detailed, in-depth discussion of computer ethics, network etiquette, policies that regulate the department where the course is taught, and computer crime laws.

Hands-on Experience and the root Problem

A hands-on, practical course on network security and intrusion detection cannot be taught with just a textbook, a blackboard, and a few slides. When teaching practical security it is necessary to allow the students to experiment with the security techniques covered in class. The main problem is that most security experiments require privileged access to the operating system – in UNIX lingo, root privileges.

Departments very seldom provide instructional labs where students (even graduate students) have privileged access and can experiment with security tools and techniques. This is mainly because of the possible dangers associated with such activities and because of the complexity that arises in managing an infrastructure with these characteristics. This situation was also the status at the Computer Science Department of Santa Barbara, where no such educational tool was available.

For the first edition of my class on network security, I was able to obtain the permission of the department to create an instructional network testbed where students were allowed to experiment with various types of attacks and defense techniques without disrupting the normal educational activity. Unfortunately, the department resources were limited and, therefore, it was not possible to take away from existing educational laboratories the hardware and software needed for a testbed network. However, the department at that time had gone through a major renovation of its computer network. Therefore, I was able to build a testbed network using pieces of

hardware that were discarded during the upgrade. The resulting testbed was composed of ten hosts (PCs and Sun Workstations) configured with a number of different operating systems. The testbed (named 'the playground' by the students) was accessible remotely and secured by a firewall, so that only authorized users could use it. This network proved to be an invaluable educational tool: For the first time the students were able to test security tools and attacks in a 'safe' environment. The feedback from the students in the class was overwhelmingly positive, and for both the second and third instances of the class the department provided support to improve the testbed network.

Even though the creation, configuration, and maintenance of the testbed required a substantial amount of additional effort from both the instructor and the teaching assistants, this experience proved that it is actually possible to create an instructional tool where students are able to enjoy a hands-on experience with security techniques.

Evaluating the Students with Live Exercises

Even though the testbed network was an important instructional tool, it didn't provide a realistic experience of the attack/defense process. Each tool and technique was experimented with in an isolated way. In addition, there was the need for evaluating the skills that the students acquired during the class.

Therefore, in the first edition of the course I decided to create a live exercise that would give the students a feel for the difficulties of both attacking and protecting computer networks in real time. During this exercise, which was conducted at the end of the class, the students were divided into two teams. The teams had to perform a coordinated attack and defense process against each other, within a limited time frame (around four hours). The students had to provide a report of their activities that allowed the instructor to evaluate their ability to put to work the techniques learned in class. The enthusiastic response of the students convinced me to include this type of exercise in every future edition of the course.

Road-map

Network testbeds and live exercises are invaluable educational tools for a course on practical security. However, they also are incredibly difficult to setup and manage and there are many lessons that were learned from conducting these activities. For each edition of the class, the lessons learned suggested improvements to the structure of the testbed network as well as modifications in the organization and execution of the live exercises. This paper describes in detail the testbed networks used during the course and the structure and execution of each live exercise, explaining the rationale behind them and discussing the lessons learned. The intended audience for this paper is higher-education instructors that may want to use testbed networks as an educational tool and reproduce similar exercises.

The rest of this paper describes the testbed networks and the live exercises used in the first three editions of the class. In the next section, a first simple testbed network is described. Then, a classic Red Team/Blue Team exercise that was run on that network is presented. The following two sections present an improved testbed design and the evolution of the first exercise into a 'capture the flag' contest. Then, two sections detail an improved testbed and a network-based 'treasure hunt', which were used in the third edition of the class. Then a section contains the discussion of some related

coursework. Finally, the last section draws some conclusions and outlines future work.

A First Testbed

The network testbed used in the first edition of the class was very simple. The topology of the testbed was flat, with a single Ethernet hub connecting a number of hosts. At that time, it was decided to use Linux RedHat 6.2 and Windows 2000 for the PCs and Solaris 2.5 for the SparcStations. A whole class C IP subnetwork (254 possible hosts) was dedicated to the testbed. It was particularly difficult to obtain from the department administrators a range of routable IP addresses, given the scarcity of IP addresses that almost all Universities are experiencing.

The network was separated from the Department network by a firewall. This allowed for careful filtering of traffic from and to the outside. In particular, the testbed network was accessible only from three instructional labs used by the students. Traffic to and from the Internet was blocked.

The students were asked to choose testbed account names that were different from the ones used in the Department network. This was done to avoid confusion between the activity performed on the testbed and the normal use of the department network. The students were given the password of the root account on the testbed machines (with the exception of the firewall).

In order to monitor the usage of the network the students were requested to send an email message to the instructor before starting to use the testbed and after having used it. The format of the email message was defined precisely, and contained the time at which the 'session' started/finished, the username and testbed handle of the user, and the IP address of the host the user was connecting from. For example, user Mark Twain (user name `mtwain` with testbed handle `bzero`) connecting to the testbed on January 31st, 2001, at 8:32am from a host with IP address 128.111.48.69 would have sent a message to the instructor with the following subject:

```
TESTBED USER bzero BEGIN 0101310832 mtwain 128.111.48.69
```

A similar message would have been sent when the session was terminated. The use of these messages allowed the instructor to devise scripts that assessed the testbed overall usage and, to a certain degree, to verify that the connections from the outside were associated with legitimate use of the testbed network. Note that this accounting mechanism was not bullet-proof and was created mainly to gather data about the students' usage of the testbed.

During the course of the class, several lessons were learned. First of all, even though the flat topology of the network is easy to build and manage, it does not allow one to compartmentalize different parts of the network. This proved to be a problem during the preparation of the Red Team/Blue Team exercise (see next section), when the class was divided in two competing teams. In some cases, the activity of one team interfered with the productivity of the other team and this sometimes prevented the students from using the testbed to its full extent.

A second important lesson was learned when particularly 'destructive' attacks were tried by the students. Even though the students were told to be very careful in controlling the execution of attacks, in some cases the attacks permanently modified the operating system of the testbed hosts, making the attacked host unusable. In these cases, the operating system had to be reinstalled from the original distribution. This activity was particularly time consuming.

Another unexpected problem was the use of the department network passwords within the testbed. If a testbed user opened a connection from a testbed host to a host in the department network and the login procedure required a password, then the password could have been captured by a number of means. Therefore, the students had to be made aware of the problem of typing passwords in any environment where the keystrokes could be logged by a number of means including kernel-level logging.

A Red Team/Blue Team Exercise

This exercise was carried out during the first edition of the course. In this exercise, the class was divided into two teams: the Red Team and the Blue Team. The Red Team was responsible for attacking and compromising a set of hosts, while the Blue Team was responsible for detecting the attacks and, in a limited form, for protecting the hosts.

The final goal of the Red Team was to obtain a file named `secret.txt` stored on each victim host. There could be multiple copies of the file and decoy copies could also be present. The only files that had to be retrieved were those whose contents started with the keyword `SECRET`.

The goal of the Blue Team was to detect the attacks coming from the Red Team. In addition, the Blue Team could execute some counter-measures to slow down or confuse the attackers. In particular, the Blue Team could freely decide where to store the secret file. The only requirement was that the file be on a mounted file system.

Some rules were introduced in order to make the exercise more interesting. First of all, the Blue Team could not filter or block any network traffic. Second, the Blue Team could not patch any vulnerability: The Blue Team had to work with out-of-the-box operating systems. These rules were imposed to prevent the Blue Team from completely patching and locking down the systems. Even though in real-life situations network access to sensitive services is actually heavily filtered, in this case a network filter and the patching of known vulnerabilities would have made the whole exercise uninteresting.

The Read Team also had some limitations. First, the Red Team could not use *a priori* knowledge about the victim hosts. It was clear that some of the hosts in the class testbed would have been used as victims. The students were invited to avoid any use of 'testbed-specific' knowledge, e.g., the association of a certain Ethernet address with a certain host in the testbed network. Second, the Red Team could not disrupt services, bring down hosts, and delete files. This rule was introduced to avoid actions from the Red Team that would have jeopardized the effectiveness of the detection tools of the Blue Team.
Participation in the Blue Team/Red Team exercise accounted for 20% of the final grade. The students had to break into sub-teams with specific tasks. At the end of the

exercise, each team had to submit a report. The report format was specified in detail so that the instructor could evaluate a number of parameters, such as the ability to plan in advance both attack and defense, the ability to deploy protection/detection mechanisms and to prepare automated attack scripts, the ability to cooperate with other sub-teams, and the ability to maintain a log of the activities (both attacks and detections).

Setup

The setup for this exercise required the configuration of two sets of hosts, one set for each team. Both teams needed root access to the hosts in order to set up attack and defense tools. The teams were told to prepare and test their tools on the class testbed network and to be ready to move their tools to different hosts right before the exercise. This was done to push student to develop portable software.

It was decided that the Blue Team hosts would be four of the hosts in the class testbed. The IP addresses of the hosts were changed, to make identification of the hosts not completely trivial. In addition, the operating systems on these hosts were re-installed to avoid the possibility of Trojan-ed software left by components of the Red Team. The network was instrumented so that a complete dump of the traffic could be collected.

The Red Team was given privileged access to a set of hosts located in an instructional lab, where the exercise took place. These hosts were the main concern for the administrators, because the students could use their privileged access to attack other hosts in the instructional lab and access the departmental file server. It was decided that the advantage in terms of management overcame the risks, and that the students could be trusted (at least for a four-hour period).

Execution

The exercise included a two-hour preparation phase, where the two teams set up their tools, and a two-hour execution phase, where the actual competition took place.
The day of the exercise, an instructional lab was completely reserved for the exercise. The room was divided into two zones, one for each of the teams.

The preparation phase was carried out without surprises. The testbed hosts were made accessible to the members of the Blue Team, who installed their tools and decided the location of the secret files. The Red Team installed the attack tools on the hosts that were placed in the instructional lab.

When the actual attack phase started, the atmosphere in the lab heated up. The students were very excited and there was a general feel that a competition had started. The competition was not just about getting a good grade in the class. The students actually felt that they were part of a team, and they had a sort of team pride.

The Blue Team had developed a number of network-based decoy tools, which were supposed to confuse the adversaries. These tools were simple but very effective. They ranged from sniffers that would respond to ICMP requests even when directed to non-existent hosts, to tools that would simulate the existence of multiple hosts by 'mirroring' the behavior of one. In addition, the Blue Team created host monitoring software that acted as a form of host-based intrusion detection.

The Red Team also developed a number of tools. Most of them were filters to translate the outputs of scanning tools into a format that was usable by tools developed by other team-mates. The attack process was organized in detail: the attackers had an 'attack pipeline' where the results from one team were given as input to the following team in a continuous process.

During the execution of the attack a few incidents occurred. A couple of times the scanning activity of the Red Team crashed a victim host. The hosts were then rebooted and restored. In a small number of instances the monitoring systems developed by the Blue Team overloaded the monitored hosts to the point that they were unresponsive and, in two instances, they had to be rebooted.

Apart from these events, the exercise progressed smoothly. The Red Team was able to successfully compromise all the hosts and access the secret files. Most of the attacks of the Red Team were successfully detected by the Blue Team. In addition, the decoys and the defense tools developed by the Blue Team successfully slowed down the attackers.

Lessons Learned

- Having a team that is responsible for defending only and a team that is responsible for attacking only has a number of disadvantages. First of all, the members of the defense team think they are having 'less fun' than the members of the attack team. In addition, they feel that protecting and detecting requires much more work than attacking. This last observation was confirmed by comparing the tools developed for the exercise. The Blue Team developed tools that were much more sophisticated than the Red Team tools. This is mostly because of the restrictions imposed on the defenders in terms of network filter configuration and OS patching.
- The development of original tools should be required, or at least rewarded more. The Red Team members downloaded most of their attack tools from the network and concentrated most of their efforts on coordinating the activities of different sub-teams. It would have been preferable to have more of the Red Team's effort devoted to developing new attack tools.
- It is necessary to specify a precise format for both the description of the attacks and the detection logs. The reports from the students contained very imprecise descriptions of both. Often, basic information (e.g., correct timestamps and TCP ports involved) was missing. This made it impossible to correctly match the descriptions of the attacks performed by the Red Team with the detections reported by the Blue Team. In addition, no automated processing was possible.
- It is important to stress the importance of a *process*. Students tend to take shortcuts (e.g., an attempt to run a known exploit blindly against the 255 addresses of a subnet) in order to win the competition. Instead, it is important to foster the preparation and the execution of a well-defined process.
- It is important that the two teams work in different rooms. Having the two teams sharing the same lab space causes a number of problems. First of all, some of the students' energy is devoted to checking if some members of the other team are trespassing. Second, noise and cheering from a team may disturb or irritate the other team.

- The Blue Team and the Red Team need to be on different IP subnets. This makes management and filtering simpler. In addition, by having attackers and defenders separated by intermediate routers it is possible to create a more realistic setup.

A Structured Testbed

Our experience with the flat testbed topology used in the first edition of the class showed that it was necessary to include hardware mechanisms to be able to partition the users of the testbed network. For this reason, the second edition of the class included a redesign of the testbed network. Given the success of the previous edition of the class, it was possible to obtain some funding from the Department to build a new (and better) testbed network.

The structure of the testbed included two separate subnetworks connected by a router. This allowed us to assign separate network addresses to separate teams, solving the problem of interference during the testing phase. The use of a router that separates the two subnetworks also allowed us to use specific firewall rules to limit (and log) the amount of interaction between the two subnetworks. The router was implemented as a multi-homed host connected to each separate subnetwork and to the outside world. The host used Network Address Translation to relay internal traffic to the outside. By doing this, it was possible to use a single routable IP address for the whole testbed, which simplifies considerably its management.

Another innovation in this testbed was the use of an image server to allow for the automatic reinstallation of OS platforms. Images of freshly installed operating systems were saved on a dedicated server. Whenever a reinstallation was needed a host was rebooted using a special disk containing the restoring software, namely Partition Image (Dupoux 2003). The software would connect to the image server and restore a clean image of the operating system in few minutes.

Even though the use of an image server simplified the procedure necessary to restore malfunctioning operating system, and the partitioning of the testbed into two subnetworks created less interference between the class teams, as the time got close to the deadline for the live exercise, the teams started to push the testbed network to its limits. In particular, each team used the hosts in its own subnetwork as both attackers and victims. This situation caused an increase in the number of incidents that prevented the testbed from functioning properly.

A Capture the Flag EXERCISE

This exercise was carried out as part of the second edition of the class. The goal was to modify the Red Team/Blue Team exercise to take into account the lessons learned in the previous editions of the class.

The exercise was organized in a way similar to the Red Team/Blue Team exercise, with the difference that there was an attempt to balance the attack and defense responsibilities between the two teams.

The class was divided into two teams: The Alpha Team and the Omega Team. Both Teams were responsible for both attacking the other team and defending their own assets. More precisely, each team was responsible for protecting a set of hosts and

hiding a flag (the secret file described in the Red Team/Blue Team live exercise) on every host. The team's goal was not to prevent the other team from breaking into the hosts. Instead, the priority was to detect the attacks of the opponents. In addition, each team had to attack the other team's hosts and retrieve the flags for each of the attacked hosts.

The rules that were imposed to the two teams were similar to those described in the previous exercise: the teams could not use *a priori* knowledge about the testbed network; the teams could not disrupt services, bring down hosts, or delete files; they could not filter/block network traffic and/or patch vulnerable software.

Participation in the 'Capture the Flag' exercise accounted for 20% of the final grade.

Setup

The setup for this exercise was different with respect to the original Red Team/Blue Team exercise. Two different instructional labs, one for each team, were reserved for the exercise. The labs were on different IP subnets. Two sets of hosts different from the ones used for the class testbed were prepared and configured in an identical way. In addition, it was decided to connect all the hosts to a hub and to provide extra connection ports for the students' personal laptops. This way they could pre-install some of the attack/defense software prior to the exercise. A complete dump of the traffic directed to the victim hosts was collected.

Execution

When the exercise started, each team gathered in their assigned instructional lab. Then, each team was given the hosts to be protected.

At the beginning of the exercise, the teams had two hours of 'truce' to prepare their hosts for the exercise (installation of attack/defense software, hiding of the flags, etc). The truce was actually enforced by a set of rules in the router connecting the two instructional labs. The actual exercise was carried out in the following two hours.

This time the students were strongly encouraged to develop their own tools. The results were impressive: the students created complete honeypots using virtual machines (e.g., User Mode Linux and VMware) and built very complex attack tools to improve the resilience to decoy techniques.

Lessons Learned

- It is important to push the students to be precise in identifying their targets. This is mainly to prevent attacks from getting out of hand, but also to make them understand the subtleties of stealthy attacks.
- Collecting data during the attack is an important activity. Extra effort should be devoted to collect host audit trails. These are particularly valuable for use in future editions of the class (e.g., audit trail analysis assignments) and as research data.
- The creation of unnecessary traffic during an exercise should be penalized. By penalizing the excessive generation of traffic it is possible to prevent the students from launching massive denial-of-service attacks against the opponents'

hosts and force them to use advanced techniques that use the least amount of traffic.

- It would be beneficial if the students were required to proceed through a path that would force them to progressively make their way to a complex network. The Red Team/Blue Team and Capture the Flag exercises had a 'flat' structure: the same techniques were applied iteratively to a number of targets and there were no changes in the mission's goals during the exercise.

A Network with Victim Hosts

The lessons learned from previous editions of the class suggested an improvement of the testbed network. For the third edition of the class it was decided to add to original testbed configuration a subnetwork that would contain victim hosts only.

Therefore, the new testbed featured a multi-homed host connected to three internal subnetworks and to the outside. Two of the subnetworks were allocated to each of the class teams. A third subnetwork contained a set of hosts that could be used as targets for attacks.

The victim machines were configured in a fail-safe mode, where a process would periodically check their availability, and, in case of malfunctioning, it would restore a clean installation of the operating system and it would execute a reboot.

The rules of the firewall allowed traffic between each of the two subnetworks allocated to the teams and the target subnetwork, but allowed no traffic from the 'target' subnetwork to the outside. This allowed for a more isolated environment for the testing of security attacks, without the possibility of interrupting the research activity of other students and limiting the interference between the two class teams. We found that this testbed configuration provides the best tradeoff between functionality and ease of management

A Treasure Hunt Exercise

In this exercise, executed as part of the third edition of the class, the Alpha and Omega teams competed in a treasure hunt. The treasure hunt goal was to break into a simulated (yet realistic) payroll system and perform a money transfer transaction.

Each team had to perform a number of tasks (e.g., scan a network or break into a host). Each task had to be completed in a limited amount of time (e.g., 30 minutes). The first team that achieved the task got 5 points. If the other team completed the task within the specified time, it received 3 points. If the time elapsed and the team was not able to complete the task, then a cheat-sheet was provided so that the task could be completed, but no points were given. A task was disclosed only after the previous one was completed by both teams. The list of tasks is presented in Table 1.

In this exercise, no detection task was required (an exercise similar to the "Capture the Flag" described previously was carried out early in the quarter as a form of midterm.). The teams had to concentrate on attack techniques only. The goal was to motivate the students to be prepared for the unknown and to be able to deal with unforeseen problems. In addition, a considerable amount of stress was put on the production of truth files, that is, files that contain a complete specification of the attacks that were

carried out. These files had to be produced in IDMEF (Curry 2003) format, for automated processing.

In preparation for the exercise it was suggested that each team build expertise in a list of topics: network scanning techniques, attacks against SQL servers (both local and remote), NIS-based and NFS-based attacks, buffer overflow attacks (both local and remote), privilege escalation techniques, password cracking techniques, attacks against Apache web servers, attacks against CGI applications. This list was provided to focus the energy of the teams on techniques that would be useful during the exercise.

Task	Description	Max Duration
1	Determine the active hosts in subnet X.Y.Z. Also determine each host's OS and the services/applications that are remotely accessible. Scanning techniques that will evade detection by the Snort system will receive additional bonus points.	20 minutes
2	Get interactive access to the web server host by exploiting a web-based vulnerability. You must be able to login into the host as a user account other than root.	30 minutes
3	Get root privileges on the web server host.	30 minutes
4	Determine the hosts that are located in the specified internal subnet. Also determine their OSs and the services/applications that are remotely accessible. Scanning techniques that will evade detection by the Snort system will receive additional bonus points.	20 minutes
5	Access the MySQL database on host SQL and obtain the content of the table Employees.	20 minutes
6	Get interactive access to the MySQL server host. You have to be able to login with an account that is not root.	20 minutes
7	Get root access to the MySQL server host.	20 minutes
8	Modify the database table Employees, setting the account number of each employee to an account number of your choice.	10 minutes
9	Obtain access to the transaction service on host TRN. Schedule a paycheck payment that will transfer the employee paychecks to your account.	30 minutes

Table 1: List of tasks used during the Scavenger Hunt exercise

It was made very clear to the students that the ultimate goal of the exercise was to perform a multi-step attack that was as realistic as possible. One of the lessons learned from previous exercises is that the data collected during these exercise is valuable from both the instructional and the research viewpoints. The traces collected in the previous exercises lacked an underlying 'plan'. That is, it was desirable to have traces

of attacks that had a well-defined final goal. This is particularly useful for alert correlation purposes. Therefore, an important by-product of this exercise was the Tcpdump data, BSM data, Windows event logs, and Snare events collected on the networks and hosts used during the exercise. Combined with the truth files produced by the students, these traces are invaluable resources for researchers in the field of intrusion detection and attack correlation (Lindskog 1999).

Setup

In order to prevent the two teams from interfering with each other, two identical target networks were setup. The topology of the networks used in the exercise is shown in Figure 1. In the following, we describe a single target network.

The web server was placed on a DMZ network. The MySQL server (host SQL), the file server (host NFS), and the transaction server (host TRN) were placed on a separate network, accessible only by the web server host.

The web server was an Apache server, running as user apache, as per default installation. In addition to a fake corporation site, a number of broken CGI scripts were installed on the web server. One CGI script was vulnerable to a phf-style attack. Another CGI script contained information about how to log into the MySQL database, namely a clear text password. A program for checking the syntax of perl files was 'erroneously' left around in the CGI directory. This program could be invoked through the web server. The program allowed one to view the source code of all the CGI scripts installed on the server, disclosing important information, such as embedded MySQL passwords.

The file server was configured to export the file system /home to the world. This is a security mistake often present in internal networks, where security is more relaxed. In addition, the host NFS was configured to provide password files through NIS.

The MySQL server provided remote access to user dbuser with password bsecret. Note that by default MySQL allows local access to root without having to provide a password. The server mounted the /home file system from the file server.

The transaction server had a service running on port 7979. The transaction application was developed *ad hoc* for this exercise by the instructor.

When connecting to port 7979, the user was dropped into a simple shell application. Typing HELP would show a list of commands, one of which is PAYCHECK. The team was supposed to invoke that function to transfer all the employee paychecks to the attacker's account. That function required a password. The encrypted (but very guessable) password was stored in the password file distributed through NIS.

This setup required a considerable effort. I developed the web site, the code for the CGI scripts and some applications, and the SQL schema for the database with the help of the teaching assistants and some of my PhD students. In addition, the networks had to be physically set up, and a whole set of services were created and configured on each network.

The setup of the exercise and the testing of the network configuration required two days of work for a team of four people.

Execution

The day of the exercise the two teams gathered in two separate (but nearby) instructional labs. The execution of the exercise included a first hour where the teams would prepare their tools, followed by a three-hour period during which the actual treasure hunt was performed.

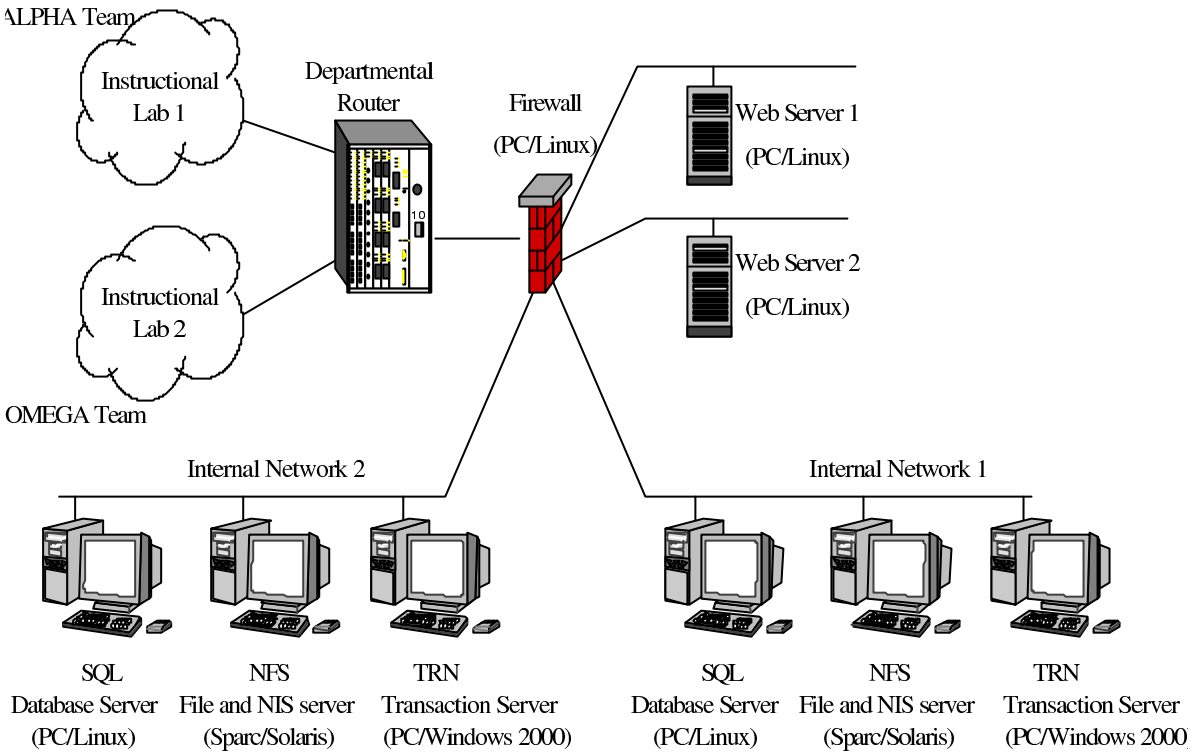

Figure 1. Treasure Hunt: Network setup

The students were extremely excited about this exercise. In this exercise there was no direct clash of teams. Therefore, there was no fear that a team could do something illegal to jeopardize the mission of the other team. By the same token, the exercise was structured as a race: the first to achieve a given task would get the most points. This motivated the students to organize their sub teams effectively.

The exercise progressed seamlessly up to task seven, where one of the team wasn't able to complete the task and had to use a cheat sheet. The same team also had significant problems in performing the final task and needed to be helped (so that the exercise could finish). These difficulties were due to the way the teams were created.

The Alpha Team was composed of students whose last names were between A and I, while the Omega Team was composed of students whose last names were between J and Z. This division didn't take into account the skills of the individual students, and by chance the most skillful students ended in the Alpha Team.

Lessons Learned

- Setting up separate targets for the two teams and having them race against each other is a very good way to foster competition without having to deal with the less pleasurable aspects of a direct clash between the teams.
- Building balanced teams is important. It promotes a fair and interesting exercise, and at the same time it supports the morale of the students by letting them know that there is not a 'best' team.
- Collecting traces of all the actions performed by the students is important. This activity should not be limited to network traffic, but it should also include host-based audit trails (e.g., Windows events).
- It is important to educate the students to create *good* truth files. These are extremely useful to identify the attacks within the logs.
- This type of exercise requires twice the effort needed to set up exercises like those described in previous sections.

Related Work

The use of hands-on experience in labs is obviously not new. Testbed networks have been previously used to provide root-level access to students. A different approach is used by a class taught in Stanford (Boneh 2002) where students can experiment with their techniques on isolated virtual machines. While this approach gives a reasonably precise idea of the working of security attacks, it is different from the 'real thing.' Only a testbed network can provide a realistic environment.

Even though hands-on experience is advocated by many, there are few graduate and undergraduate courses on computer security that offer live exercises as part of the course. For example, Georgia Tech offers a class (Santos 2002) where a team has to install a number of services on a Windows host and another team has to perform attacks. In this case, each of the two phases, preparation and attack, lasts a couple of weeks. As another example George Washington University offers a class (Daniel 2002) that includes the creation of honeypots and some sort of team-based interaction.

In general, live exercises are difficult to organize and conduct, and, therefore, instructors generally prefer other types of educational tools, which are less expensive in terms of time and hardware/software resources. We believe that our experience, especially the adoption of the treasure hunt exercise is rather unique.

Conclusions and Future Work

Testbed networks and live exercises are important instructional tools in teaching the practical aspects of network security. Testbed networks provide a safe, isolated environment where students can experiment with potentially dangerous technologies. Live exercises motivate the students to give their best because of the competitive nature of the test, and because their success is heavily determined by the students' creativity.

Testbed networks and live exercises are also extremely difficult to organize and manage. In particular, testbed networks require constant management and the implementation of effective recovery techniques. Live exercise require detailed preparation, because they are executed in a short time span and if anything goes wrong it is difficult to solve problems within such tight schedule.

This paper described the testbed networks and the live exercises that have been used as part of a graduate-level course on network security and intrusion detection. The class has a successful history of attendance (the maximum allowed is 40, but classes ranged between 50 and 90). The success of the class is determined by the practical, hands-on approach that was adopted. The student feedback was overwhelmingly positive. Some of the students that took a previous edition of the class even came to observe the live exercises of other editions.

This class has received some attention by other instructors. The class materials have always been online and have been used in other courses (with permission of the instructor). In addition, the by-products of the class attracted the interest of research groups. We are currently preparing a web site to distribute the traces collected during the different exercises, in addition to the course material. We hope that this effort will allow other courses to use our experience to build similar testbeds and live exercises.

The next step in the evolution these live exercises will be the creation of an inter-institutional 'Capture The Flag' exercise, where students that are attending similar practical security classes at different universities can test their skills against each other. This form of live exercise will be very challenging because of the high risk of creating unfair conditions and the possibility of a very high level of competition. For this reason, the careful design of the exercise is critical. We plan to use the lessons learned from the design and implementation of previous live exercises to design effective mechanisms for containment and measurement of this experiment.

References

Bishop, M. (1997). The State of INFOSEC Education in Academia: Present and Future Directions. In *Proceedings of the National Colloquium on Information System Security Education*, pages 19–33. Keynote address, Linthicum, MD, April 1997.

Bishop, M. (1999). What Do We Mean By "Computer Security Education"? In *Proceedings of the 22nd National Information Systems Security Conference*. Arlington, VA, October 1999.

Bishop, M. (2000). Academia and Education in Information Security: Four Years Later. In *Proceedings of the Fourth National Colloquium on Information System Security Education*. Washington, DC , May 2000.

Boneh, D. (2002). *CS155: Computer and Network Security*. Stanford University.

Brandt, A. (2003) Class on Virus Creation Draws Industry Ire, *PC WORLD*, May 2003.

Curry, D. and Debar, H. (2003). Intrusion Detection Message Exchange Format: Extensible Markup Language (XML) Document Type Definition. IETF Proposed Standard, Version 1.0, http://www.ietf.org/html.charters/idwg-charter.html, January 2003.

Daniel, R. (2002). *ECE 297 - Special Topics. Network Security: Honeypots*. The George Washington University, School of Engineering and Applied Science, Department of Electrical and Computer Engineering.

Dupoux, F. (2003) *Partition Image*, URL: http://sourceforge.net/projects/partimage. June 2003.

Lindskog, S., Lindqvist, U., and Jonsson, E. (1999). IT Security Research and Education in Synergy. In *Proceedings of the First World Conference on Information Security Education (WISE1)*, pages 147--162, Kista, Sweden.

Santos, A.D. (2002). *CS6265 Information Security Lab*. Department of Computer Science, College of Computing, Georgia Tech.
Schneier, B. (2003) *Crypto-gram Newsletter*, June 2003.