VASSILIS PREVELAKIS

supporting a security laboratory

Vassilis Prevelakis is assistant professor of computer science at Drexel University in Philadelphia. Over the past 12 years he has been involved in numerous security projects, both as a network administrator and as a researcher; currently, he is leading a project that aims to improve security for home networks.

vp@drexel.edu

MANY YEARS AGO, WHEN I WAS AN undergraduate student, when showing freshmen students around campus we would point to a large crater-like depression in the ground and say, "And this is the site of the old chemistry laboratory" and smile at the allusion to a horrible accident. The trick worked because people (especially freshmen) associate chemistry labs with explosions. However, running a security lab at the undergraduate level can also lead to "interesting" situations. Care must be taken so that the experiments do not disrupt the campus network or, heaven forbid, escape into the Internet.

Another question is what kind of experiments should be run so that the students derive real benefits from these labs. We do not want to teach students to become script kiddies, learning procedures by rote without really understanding what is going on. Moreover, students should have the means to evaluate their proposed solutions to problems that have been set out for them. In this way they reinforce their learning by actually putting into use concepts discussed in class. The labs, thus, do not replace the normal lectures but, rather, augment them. For the labs to be effective we need to ensure that students (a) actually spend time thinking about what they are doing rather than simply following some checklist, (b) learn concepts, rather than being trained in the use of specific programs, and (c) develop their ability to analyze complex situations and arrive at convincing solutions to problems.

At the Computer Science Department of Drexel University we have created a security lab environment and associated course work with the objective of meeting these aims. Our goal was to ensure that students could participate in lab sessions while also giving them the option of working with the security lab environment outside the lab sessions. Either way, students should be able to work independently without interfering with each other.

Experiments

Let us first discuss a number of experiments that we created for the lab and then we can describe the environment we created to run them.

We use two topologies to try out different experiments. In both cases we have three hosts (A, B, and C) that are used for the experiment, plus a fourth host (G), which connects host A to the campus LAN to allow students to exchange files with departmental servers. The first one (bus topology) is the traditional LAN layout, where all the hosts are connected on the same LAN (Figure 1a). The star topology shown in Figure 1b is mostly found in WAN situations, where A, B, and C are routers connecting internal networks together over long-distance point-to-point links

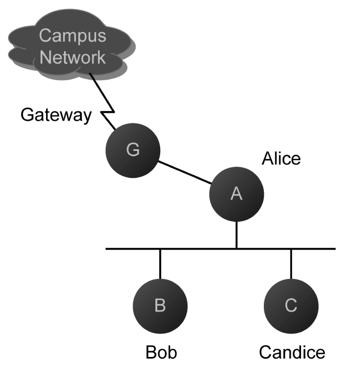

FIGURE 1A: BUS TOPOLOGY, WHERE HOSTS A, B, AND C ARE CONNECTED ON THE SAME LAN

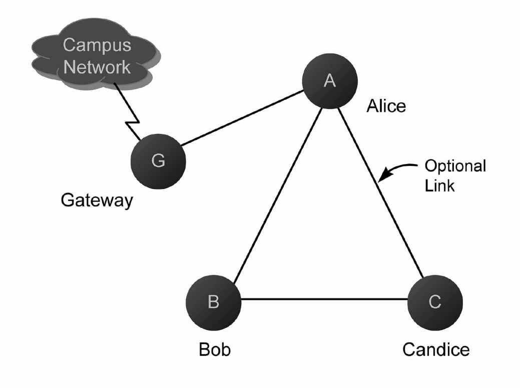

FIGURE 1B: STAR TOPOLOGY, WHERE A, B, AND C ARE EACH CONNECTED TO TWO OTHERS VIA SEPARATE LANS

EXPERIMENT 1: ARP SPOOF

In our first experiment we want to fool Bob into talking to the wrong DNS server and we do this by installing a fake DNS server on Candice and performing an ARP spoofing attack on Bob. The main purpose of this experiment is to show how protocols lacking authentication (such as ARP and DNS) can be subverted, but it also serves to familiarize the students with the ways raw packets can be generated by user-level applications.

This experiment has three stages: installing the fake DNS server, constructing and running the program to carry out the ARP spoof attack, and troubleshooting the fake DNS server in order to complete the attack. Students are provided with a simple DNS server replacement (dproxy) and they have to configure it on both Alice (the “valid” server) and Candice (the malicious, or “fake,” DNS server).

We chose dproxy because it is a very simple DNS proxy server. Although dproxy listens for DNS requests in the same way as a usual DNS server (e.g., named), rather than resolving the queries itself, it simply uses the resolver library. The big advantage of this is that dproxy can answer queries by looking at the /etc/hosts configuration file, so we can easily add a new entry (e.g., www.drexel.edu, but even one belonging to a bogus zone such as www.priv) by editing the /etc/hosts file.

Students start the experiment by setting up dproxy on Alice and setting up Bob to use Alice as its DNS server (i.e., adding Alice to the /etc/resolv.conf file). They also configure dproxy on Candice and add two bogus entries pointing to itself: one for Alice and a second using a fictitious domain (www.priv). For example, by assuming that Candice has IP address 192.168.100.3, the new entry in /etc/hosts will look like:

```
192.168.100.3 alice www.priv
```

Students then run a few queries (nslookup and ping) to ensure that they have correctly configured their machines and establish a baseline for observations. They then activate the ARP spoof program on Candice and carry out the same queries, observing that while Bob now sends its requests to Candice, the request fails because dproxy uses Candice's source IP address and not Alice's. Students have to solve this problem and run the complete spoofing operation, fooling Bob into thinking www.priv really exists.

EXPERIMENT 2: ROUTING/FIREWALL

The triangle topology in Figure 1b is used as the basis for a WAN scenario where A, B, and C are routers connected via point-to-point links.

In the routing experiment students run a routing protocol (we used RIPv2 because it is the easiest to configure) and observe how they can divert traffic to Candice by injecting routes. The fake DNS server from the first experiment is also used here to exploit the redirection.

By removing the link marked "optional" in the diagram, the same topology can be used to create a configuration where B can serve either as "man-in-the-middle" or as a firewall. In the man-in-the-middle scenario, students observe packets going through B to spy on communications between A and C. For example, we ask students to use telnet to log onto A from C, while running tcpdump on B. Then students extract the log-in password from the packet traces.

In the scenario where B is an IP firewall, students design various configurations showing how B can filter packets, perform network address translation, etc.

Making All This Happen

Running these experiments in a safe manner while allowing an entire class of students to work at the same time during the lab session has been a daunting task. About five years ago we installed a rack with about 20 computers interconnected via a large switch with more than 100 ports. Each computer had 3 or 4 Ethernet interfaces, and they were all connected to the switch. By partitioning the switch ports into independent groups (VLANs) we could create various interconnection topologies for the rack machines (Figure 2). On each machine, one of the interfaces was reserved for management, allowing network access to the machine regardless of the configuration of the other port interfaces. As a last resort, serial access to the console ports of each machine was also provided.

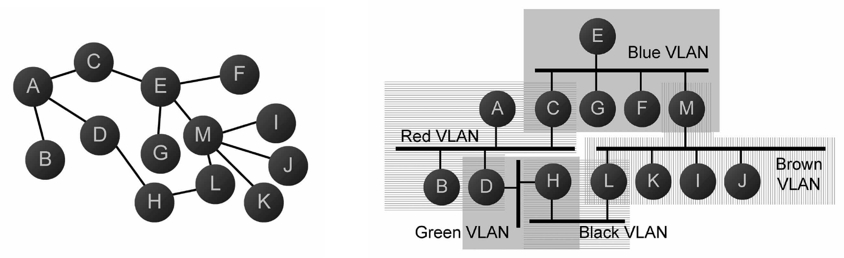

FIGURE 2: VARIOUS TOPOLOGIES (LEFT DIAGRAM) CAN BE REPRESENTED BY CONFIGURING VLANS ON THE ETHERNET SWITCH (RIGHT DIAGRAM)

However, the main problem with this approach was that reconfiguring the switch was extremely laborious and error-prone, and, to make matters worse, we had to provide special configurations for each machine.

Scripts implementing canned configurations for the switch were developed and the rack machines were configured to boot from the network and use NFS for their file systems, but still there were problems. For example, if we allowed students to have access to their home directories on the departmental server they would also have access to the files of other students. Although NFSv4 supports user authentication, the version of NFS we had at the time supported client-side authentication, so it was an all-or-nothing solution. In addition, the number of the machines was inadequate for the size of the class so we had to split students into groups. Finally, the students complained that they did not have access to the machines outside lab hours, so they could not work on their own.

VMWARE COMES TO THE RESCUE

To address the limitations of the hardware solution, about two years ago we started migrating our security lab to VMware. The students use a lab with Linux PCs connected to the campus LAN. There are sufficient PCs in the room for each student to have his or her own workstation.

The topologies described in the previous section were implemented using virtual machines (VMs) linked together via virtual networks (vmnets) that are included in the VMware product. The vmnets may be used to connect virtual machines together, to link them to the host workstation, or even to provide direct access (via a virtual Ethernet bridge) to the physical network. For the lab virtual machines we used exclusively host-only networks that do not allow direct communications with the outside network. For example, to create the layout in Figure 1a we created one host-only vmnet and linked all the virtual machines together. For the layout shown in Figure 1b we created virtual machines with two or three virtual network interfaces each and linked them together via three vmnets (one for each side of the triangle). Students use separate windows for each virtual machine and so can see output from all three VMs at the same time.

Hosts A, B, and C in Figure 1 are instantiated as separate VMs, whereas host G is the workstation hosting the VMware session. Each VM runs a complete installation of OpenBSD 3.8, allowing students to develop programs and test them in the target environment. Previously, programs that required administrator access to run (e.g., used raw sockets, low-numbered ports, etc.) could not be run in the common servers used by the department and many students could not run OpenBSD on their own PCs. This forced students to carry out program debugging during the lab sessions, which distracted them from the actual lab tasks. With VMware, students have the option of running the security lab environment on their own personal computers and can rerun the assignments on their own.

A big problem with running three virtual machines per student is that during the beginning of the class, 60 to 90 VMs are booted. Although the CPU load is not important because students are running VMware on their workstations, the file I/O load on the NFS servers is tremendous. This not only caused serious delays in the initial classes but caused many students to exceed their disk quotas as they started using the disks associated with their VMs. We addressed this problem by using a special feature of VMware called “non-persistent virtual disks.” This allows a virtual disk to be read-only, but in a way that does not cause problems with the operating system running in the VM. Normally, an operating system expects its boot disk to be writable, so simply making this disk read-only is bound to cause problems. Instead, the VMware non-persistent disks allow full read/write capability while the VM environment is running, but once the VM is shut down, all changes are lost. We can even reboot the guest OS and it will still see the modified image, as long as we do not restart the virtual machine environment. More information on non-persistent virtual disks is available on the VMware Web site [1].

Thus, we created three virtual disk images (one for each of the three machines in Figure 1) and we asked students to attach these images to their virtual machines and mark them as non-persistent. Since the volumes are read-only and belong to the teaching assistant, none of the students can attach these disks read/write anyway. Using non-persistent disks in turn necessitates providing some nonvolatile disk space that can be used by students to save their work. We addressed this issue by allowing each student to create another virtual disk, which is stored in the student's home directory and is only big enough to contain the student's personal files. The second virtual disk can be mounted at any place in the file system (both manually and automatically during boot), so its existence can be completely transparent to the student.

Another advantage of using common virtual disks for the operating system is that new VMs or changes to the configuration of existing machines can be added quickly and applied to all students at the same time. This makes it possible to create on short notice new experiments to demonstrate a new technique or to provide an example for something discussed in class. For example, in order to get students to carry out code injection attacks, we created a new VM with an old version of FreeBSD containing a number of vulnerabilities and asked students to come up with attacks. In this case, rather than all the students attacking one machine and thus potentially interfering with one another, we had each student boot a private VM with the vulnerable system and attack it at leisure.

Running the Labs

With the environment ready and the lab sessions created, a key question was whether to run them as homework assignments or as actual lab sessions. The former has the advantage that students can go through the assignments in their own time, and it also reduces the logistics associated with the lab sessions (booking a room with 30 workstations, making sure that VMware works correctly on every station, etc.). Moreover, students prefer to be able to work on the lab sessions outside the (limited) lab hours. Nevertheless, we feel that carrying out the experiments during the lab sessions is very important as students who are stuck can be nudged toward finding the solution. Otherwise, students will simply get the solution from fellow students and apply it blindly just to get on to the next question rather than solving the problem.

The next question is how to structure the experiments and, more important, how to phrase the instructions and questions to ensure that students do not simply look ahead to the next steps in order to deduce the answer to the question. This forced us to think about some way to prevent students from looking ahead before they answered the questions.

Our first approach was to use an overhead projector with slides describing one question at a time and wait till everybody had answered it before moving to the next one. This failed miserably, as students had to wait for the slowest one to finish and hence either students were bored or slow students were hurried along (or just given the correct answer so that the class could go on). Also, students could not go back and look at previous questions or review information given out earlier. We briefly tried handing out the assignments in printed form, one question at a time, but this meant that the lab assistants were spending more time distributing sheets of paper than answering questions.

Finally, we decided to bite the bullet and use an on-line course work system (WebCT). Each lab exercise is encoded as an online quiz that prevents students from changing submitted answers to questions. In this way students may proceed at their own pace, but since they receive no points for skipped questions, they have a powerful disincentive to peek ahead.

Unfortunately, WebCT is a very temperamental system with a lot of obstacles for casual users. For example, at one time, during the lab we found that the text boxes that students would use to type in their responses were limited to 100 characters. Since the lab was already in progress, we had no way (or clue) how to fix this, so students were forced to be brief. (This is not such a bad thing in retrospect, but one student remarked that the most challenging aspect of some questions was the requirement that the reply should be fewer than 100 characters.) Having used WebCT about 10 years ago, I wish I could go back to that older version, which, while lacking all the bells and whistles, actually let the user be in control.

Lessons Learned

- Using non-persistent virtual disks for booting the VMs and for storing the bulk of the data needed for their operation is a clear winner. However, there have been instances where students lost work when they shut down their VMware session without copying their work to their private persistent partition. We are investigating various techniques for reducing this risk. For example, on virtual machine B (the one used for the firewall and man-in-the-middle experiments) we have moved most of the system configuration normally stored in /etc to the private partition. Initially, students copy an existing partition (with the configuration of B) to their home directory and attach this copy as the second disk drive on host B. (OpenBSD sees this partition as wd1a.) Since they own this partition, they can make changes to it and these changes will persist across VMware sessions. Students can always return to the initial configuration by copying the shared partition over their private copy, thus destroying all the changes they have made. Of course, once students start working on their private copy of the /etc directory, we lose our ability to change virtual machine configurations globally. This is why we provide this capability on only one of the VMs.
- The extremely rich environment supported by UNIX sometimes works against us, as we cannot create a platform that contains all possible editors, shells, development environments, etc., that students are used to working with. Students thus often have difficulties because they are not familiar with a particular utility. For example, when we created the original environment, the default installation of OpenBSD did not include the emacs editor, and some students complained that they did not know how to use vi to make changes to various files. Since the lab environment is currently used only for the security course, students cannot be expected to spend much time customizing their environment or learning how to live with its peculiarities.
- Support for graphical user interfaces (GUIs) is needed. Programs such as Ethereal offer powerful ways of representing and managing captured data using a GUI. We believe that students would benefit from the use of such programs, but the current lab environment only supports char-

acter-based consoles. Allowing the X11 window system to run on the virtual machines is not difficult, but it causes a lot of headaches, such as performance degradation; worse configuration issues than those discussed earlier, as students are forced to live with potentially different window managers, X11 settings, etc.; and more lab assistant time to help students.

- As practically all students now have powerful laptops or home computers, we are considering the possibility of asking students to install the entire environment on their own computers and use it to carry out their regular assignments (i.e., use the security lab environment for all security course homework). Unfortunately, VMware currently runs only under Windows and Linux, which means that some students (e.g., Apple users) will not be able to use the virtual environment. Staff limitations make supporting multiple VM environments difficult, but we hope to be able to support Parallels on the Mac in the near future.
- Despite the WebCT-related setbacks, we find the on-line quiz format the best solution so far, but we are looking for alternatives to WebCT.

The security lab environment is a work-in-progress, as we always find things that can be improved and we constantly add new experiments or fine-tune existing ones. Despite the numerous issues we have had to address over the past four to five years, students enjoy the labs and we find (through exams and continuous assessment) that their understanding of security concepts has been improved by the lab experience. We believe that the same environment can be adapted for use in other systems courses, such as computer networks and operating systems, and we are planning to support such courses in the future.

REFERENCE

[1] http://www.vmware.com/support/gsx25/doc/disks_modes_gsx.html.