Security Exercises for the Online Classroom with DETER

Peter A. H. Peterson, Peter L. Reiher
{pahp, reiher}@cs.ucla.edu
University of California, Los Angeles

Abstract

Creating high-quality homework with an emphasis on creativity and open-ended learning is challenging. This is especially true for online classes, which must be both accessible via the Internet and comparable in quality and value to projects that could be used in a traditional classroom. UCLA recently began offering an online master's degree program in computer science, which includes a course in computer security. This motivated the design of online coursework intended to take the place of the traditional in-class homework and group projects. The resulting security labs use standard security tools and the DETER testbed, which can be organized into networks of physical machines running real software. In these environments, students perform open-ended exercises involving file permissions, firewalls, software vulnerabilities, eavesdropping and injection, man-in-the-middle attacks, computer forensics, and network intrusion detection systems. We also created an extensive online lab manual to accompany the exercises. With some important technical caveats, DETER proved to be an excellent platform for online education, and the labs themselves have been a great success since they were introduced in 2008.

1 Introduction

In 2007 UCLA created an online master's degree program in computer science which inspired the design of new online security coursework emphasizing the practice of computer security. Web-based education presents a special challenge to exercise designers, because the coursework must have the same quality and educational value as "offline" homework, and must be easy to access and use without much face-to-face interaction. Brick-and-mortar graduate-level CS courses at UCLA typically include a quarter-long research project, often including regular discussions with the professor during office hours, progress reports and presentations. While some of this communication could take place over video chat, the online master's program uses prerecorded lectures and is thus attractive to students who work full time and may not have compatible schedules. Because of this communication obstacle, it would have been easier to use more traditional pen-and-paper homework for the online course.

However, we felt that "hands on" labs were essential. Computer security is only successful when proper theory is married with correct practice, and the practice of computer security is tied to the real world of software and hardware. While the theory itself can be straightforward, applying that theory with the insecure protocols and systems we actually have can make engineering end-to-end security quite challenging. Students who leave a security course with an understanding of both theory and practice are have advantage over students who only understand one or the other. To achieve this, we wanted the labs to reinforce the theoretical security curriculum with exercises using current software and hardware.

We also wanted the labs to be topical and engaging. Part of what makes computer security interesting is the social, political and economic significance of security issues in today's world. Accordingly, we tried to use interesting and realistic scenarios in the labs that would engage students' imaginations so that they could see how their knowledge and actions could affect security in their own research and work.

We could have used virtualization instead of a testbed to support these labs. Free, high-quality virtualization platforms have allowed many traditional computer science courses to use real operating systems and software within virtual machines, rather than using simulators or toy operating systems. However, a number of issues with virtualization led us to use the DETER testbed as our exercise platform.

The DETER testbed is a collection of physical machines and programmable routers that can be configured dynamically and accessed via the Internet. Our labs cus-

tomize DETER Linux disk images to create live security exercises involving realistic scenarios. All portions of the labs were created using popular and widely available open source tools, several of which are canonical tools for security work. This infrastructure is accompanied by an extensive lab manual with background information, engaging scenarios, links to online resources, and short introductions to the relevant software tools.

2 Why Not Virtualization?

It would be tempting to use virtual machines such as QEMU,[1] Virtualbox,[2] or VMware,[3] as the platforms for our exercises. Under such an approach, virtual machines (VMs) would run on our students' personal computers and would reproduce experimental environments by running the actual system software in question. In order to do this, the students would download the virtualization software, virtual machine images, and perform most lab work within those virtual machines. Several popular computer science courses at UCLA, such as the undergraduate and graduate operating systems courses and the undergraduate databases and web applications courses, use virtual machines to do just that. However, there are a number of reasons why we felt that virtual machines were not the best choice for our exercises, some of which are outlined below.

First, some of the labs require multiple hosts. For example, the Man-In-The-Middle (MITM) lab requires four independent hosts: `alice` and `bob`, an eavesdropping host, `eve`, and a simulated Ethernet switch. This would require running up to four independent VMs plus the host OS simultaneously. This represents significant resource consumption, even for modern machines.

Additionally, the download for the labs discussed in this paper totals approximately ten gigabytes uncompressed, not including VM software. This includes five or more system disks, three additional hard drive images (for the Forensics lab), and all necessary software. While the courseware could be written to carefully reuse all common data, doing so could produce fragile software and be costly to develop.

Next, there is often a significant support cost to using virtualization for class projects — even in the traditional classroom. One virtualization system can change significantly between versions, and differences often crop up between various platforms for the same VM software. Also, multiple virtual machines, as used in our labs, require virtual networking or virtual network hardware. For example, the MITM lab is based on ARP-poisoning [3] and thus requires a virtual link layer. Virtual networking configurations can easily change between versions of the same virtualization system, between platforms, or even between kernel versions.

Another reason we felt VMs were less suitable is that they would be difficult to change in the event of bug-fixes to the lab software. Each lab is essentially a large, complex piece of software, and mistakes or ambiguities continue to appear even after having run the labs several times. Students using VMs would need to download and apply fixes individually and without assistance. TAs would need to ensure that students were running the latest versions of all software. Ultimately, we do not want to take on the responsibility of remotely supporting virtualization software for users.

Finally, a virtual machine running on a student's computer has no protection from snooping or tampering, since students have full control over their own machines and the VM disk images. This facilitates cheating because it is not possible to keep secrets (such as source code, directory locations, or login information) from students if they have all the raw materials; they can simply mount the disk images and look for the answers.

3 DETER

DETER[4] (for cyber-DEfense Technology Experimental Research laboratory) is a medium-scale testbed with about 300 nodes [1]. DETER is run jointly by USC ISI and UC Berkeley and is based on the University of Utah's Emulab project [4]. Many testbeds allocate machines for parallel processing or simulation. However, DETER and Emulab are dedicated to the dynamic construction of physical computer networks through the use of programmable routers and dedicated physical machines. Much of the material describing DETER in this paper also applies to Emulab, although DETER is designed for security experimentation specifically and includes stronger firewall policies and other security features.

DETER's simple configuration language makes it easy to define "experiments" — distinct configurations of physical machines and networking resources drawn from the testbed's pool. Networks can range from tiny LANs to WANs, with adjustable characteristics such as bandwidth, loss, and jitter. Users can load their own disk images on physical computers, or choose from a library of operating system images, including Windows, Linux, and BSD. Experimental nodes can be further customized with scripts that run at first boot, installing software from NFS file systems.

DETER is the ideal platform for our labs. Students acquire individual logins for the testbed, and the labs are set up as DETER experiments within our `UCLAClass` group. It is centralized and managed via a web interface, there is ample storage space for project software, and students can be assigned their own logins and private storage. DETER has about 300 machines available

for use; as a result, a class of 30 students can work simultaneously on the largest lab (using four machines), each with their own experimental nodes, using less than half of DETER's resources. Because the labs run on DETER, students do not need to download, install, or manage any local software beyond an SSH client and web browser. This allows them to work on homework from virtually any Internet-connected computer in the world.

Similarly, because the software is centrally located, fixes or other changes to the labs can be applied once and be rolled out to all students. The underlying platform and machines are fairly stable, and DETER has its own testbed operators dedicated to keeping the system running. As a result, most testbed-related issues can be resolved via email within an hour. DETER also provides backups, which protects student work and the labs themselves from catastrophic loss. Finally, the labs can be easily reused simply by instantiating new users and experiments.

4 Projects and Lab Manual

Because these labs are used in an online course, we need to provide students with high-quality, thorough information they can access remotely. We also need to be able to update the material easily, require permissions on the sections so that we can "unlock" the lab manual progressively, and want the ability to cross-link related information in the manual. We chose to use the MoinMoin wiki[5] as our content management system for its simplicity and ease of use. All lab documentation was written into and is provided through the wiki, which is configured to be read-only for the students. The top level of the lab manual contains links to DETER instructions, a document (LabTools) containing tutorials for all software used in the labs, external resources, and lab-specific pages. A link to our lab manual is provided in the notes of this paper.[6]

Each lab has its own manual which includes necessary background information, such as a technical discussion geared towards the lab in the context of computer security, links to external resources, and a link to our "LabTools" document describing the recommended software tools for each lab. Following the background information is a narrative section, "The Story So Far..." which describes and motivates the lab scenario. Each lab-specific manual closes with the list of assigned tasks and any DETER-specific instructions.

The first lab is an introduction to DETER and the UNIX command line, and subsequent labs are sequenced to build on the skills learned in earlier labs. For example, the file system Permissions lab, which introduces the concept of suid-root executables, is followed by the Exploits lab, which includes insecure programs with suid-root permissions. The Permissions lab assumes very little past experience, while the final lab, covering network intrusion detection systems, assumes the experience that should have been gathered over the course of the quarter. Accordingly, the lab manual is revealed section-by-section over the quarter. This serves to keep the students from being distracted by extraneous information, and, since the labs build on one another, allows us to refer back to previous material, or restate answers from earlier labs in later material.

The following sections briefly describe each lab, including the tools used, assigned tasks, interesting outcomes, and with the exception of the Intro Lab, a quotation from the introduction to the assignment.

4.1 Intro Lab

The Intro Lab is not a full assignment in and of itself, but serves as an introduction to DETER and basic UNIX command line skills. This "mini-lab" encourages students to set up their DETER accounts, experience connecting through the DETER firewall, and use a command shell on an experimental node for the first time. Instead of a narrative, there are four short assignments: swap in a DETER experiment, use tools like `find` to locate a set of files, research answers to several basic UNIX questions online, and submit an archive of the results.

4.2 Permissions

> *You are Wilbar Memboob, the security administrator of FrobozzCo. You are looking to hire a new system administrator to replace the guy you just fired. Unfortunately, even though his resume looked great, once you sat him at a console, it was clear he had no idea what he was doing. ... To keep this from happening again, you've created a test for new applicants. However, before you can grade the tests, you need to create an answer key.*

The Permissions lab covers complete POSIX file system permissions including setuid, setgid and the special permissions. The lab also includes access-related issues such as the `sudo` utility, POSIX login process, shell escaping, and interactions between these systems. Stateful firewall basics are covered using `iptables`-based firewalls. Key concepts for this lab are "the principle of least privilege" and "deny by default" design.

The software tools used for this lab are all straightforward, including the standard utilities such as `chown`, `chmod`, and `adduser`. Students also modify `/etc/passwd`, `/etc/group`, and `/etc/shadow`. The software used for the firewall portion is more interesting, including networking tools such as `nmap`,[7]

`telnet`, and `netcat`[8] for testing the user-created firewalls.

Our challenge in designing this lab is to provide realistic, but interesting problems that are still accessible to students — most of whom are completely unfamiliar with access control issues and permission models. The challenge for students is to consider the extended ramifications and complex interactions of these systems.

4.3 Exploits

> *FrobozzCo is a large corporation with a great many secrets. ... Unfortunately, it is clear that someone has "rooted" the server; a number of root access files were copied out over the Internet. Fortunately, no data was destroyed, but the intruder had full control of the server. You are convinced there is an exploitable buffer overflow bug in the web server software, but your boss, John D. Flathead III, laughed off your suspicions saying, "I wrote the web server software – and I'd never make that mistake!"*

The Exploits lab introduces students to buffer overflows, pathname attacks, and SQL injections. The exploits are exemplified in three applications: the buffer overflow is found in a webserver written in C and causes a segmentation fault, the pathname exploit is found in a Perl content management system and allows HTTP access to `/etc/shadow`, and the SQL injection allowing full access is found in a PHP/MySQL banking application. Each application consists of no more than a few hundred lines of code.

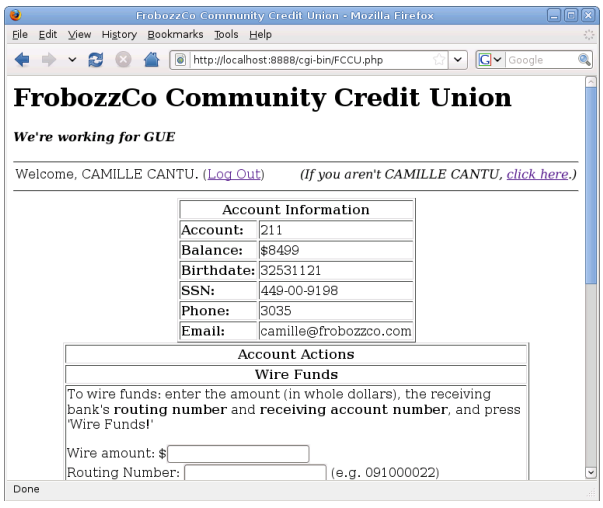

Figure 1: FCCU is vulnerable to SQL injection attacks.

The students must find and exploit the flaw in each application. For example, in the pathname exploit code, this entails forcing a web application (which is inappropriately setuid-root) to display the local shadow file. After demonstrating the flaw, the students create a patch to close the vulnerability (including redesigning the Perl script to avoid setuid-root) and write a short memo describing their work. An important security concept at the heart of this exercise is the false nature of "security by obscurity."

Studying common vulnerabilities is useful because we feel that in addition to understanding how exploits work on *paper*, it is important for students to experience the end-to-end process of leveraging small vulnerabilities into large breaches, as well as analyzing and remediating flaws. With pathname and SQL injection vulnerabilities in particular, our labs demonstrate that vulnerabilities are not due to weaknesses in POSIX permissions or SQL, but in the applications that use those systems.

Students use `gcc`, Perl, MySQL, PHP, `patch`, and several standard UNIX utilities in the course of completing this lab. Students are encouraged to use port forwarding (via an SSH tunnel) so they can interact with the web applications directly from their own desktops. This provides essentially the same usability that the student would have if the server was on a local network.

Students continue to find new exploits for all applications and develop creative patches. Due to the workload and overall difficulty of this lab, remote execution of the buffer overflow to gain root access is an extra credit challenge problem. In the future, we would like to place more emphasis on ideal remediation techniques and a deeper understanding of software vulnerabilities.

4.4 Forensics

> *Episode 12: "POST Mortem" — It was a pretty typical morning, all things considered. One client was looking for a binary bloodhound to find out whether his server was infected with a worm. The second client had some sensitive data stolen and wanted some help interpreting the evidence. The third client was a little more interesting – it was a high-rolling fat cat who was being extorted by black-hat weasels from some remote corner of the Internet. Typically, the only information available was on the computers themselves...*

Computer forensics is an exciting aspect of computer security – somewhere between CSI and Sherlock Holmes. Due to the increase in the use of networked computers and the related increase of computer theft and crime, computer forensic science is a growing field. But

even without computer crime, good security engineers must, at times, put on their "detective cap" to figure out why a system failed. This lab provides students with a truly open-ended environment where their task is to recover forensic evidence and draw conclusions from it.

While these labs were not created for a course on computer forensics, there are a number of related ideas we wanted to address, including chain of custody, cryptographic hashing for integrity, block-level disk copying, recovery of deleted files, log analysis, and places where data is often overlooked (e.g., swap disks and caches). These are key issues for both the computer forensic scientist and the security engineer.

The students use common tools which are all available under open source licenses, such as hexadecimal editors (`hexedit`), disk imaging software (`dd`), cryptographic engines (`gpg`, `md5sum`), rootkit detectors (`chkrootkit`),[9] password crackers (`john`),[10] undeletion utilities (`e2undel`),[11] and so on.

Creating the disk images for investigation was the most time-consuming task for this lab. Since the images are meant to be legitimate forensic images that can be freely interpreted by students, each image must have a minimum of setup artifacts. In order to do this, we installed a base image on a computer and actually performed the "attacks" in each scenario so that the systems would have authentic and accurate logs. After this process, minor details were changed, such as IP addresses and host names, using a script that left modification times and other meta data unchanged. We discuss future work on this lab in Section 7.

4.5 Man-In-The-Middle

> *Everyone thinks you've been hired at FrobozzCo to replace an employee that recently quit. In reality, you've been hired by IT to perform a Red Team exercise against the network. The top brass are scared of disgruntled employees following a recent online bank heist, so they've ordered a vulnerability analysis. How much damage can a sufficiently motivated employee do?*

A "Man in the Middle" (MITM) attack refers to a particular kind of attack against encryption, where an eavesdropper exchanges the public keys of two communicating parties for keys of her own creation. This can occur without the knowledge of the communicating parties, and in this way, the eavesdropper can intercept and even change encrypted communication without the parties' knowledge. Of course, doing this requires the ability to eavesdrop on the communication.

While Ethernet switches are more secure than hubs, the isolation provided by switches is weak due to the insecure Address Resolution Protocol (ARP). ARP is used to associate Medium Access Control (MAC) addresses with IP (Internet Protocol) addresses on a LAN. ARP is necessary for Ethernet frames to be delivered to the appropriate hosts. By subverting ARP [3], any host on a typical switch is able to impersonate any other host, including the network gateway. As a result of this impersonation, the Ethernet switch will dutifully redirect packets to the attacker. From this point, the attacker can even forward the packets to the true gateway, covering her own tracks.

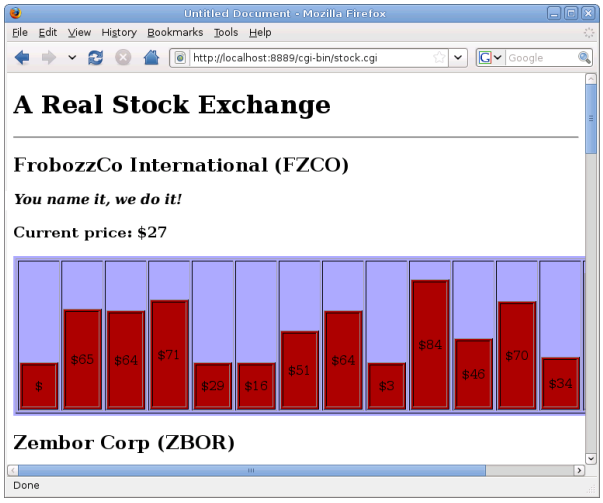

Figure 2: Students attack this application by modifying plaintext output and by forging nonces to insert bogus data.

The assignment portion of this lab features MITM, eavesdropping, ARP poisoning, and injection attacks against network services (such as HTTP, HTTPS, and telnet) using the powerful attack tool, Ettercap.[12] Students also use `tcpdump` to capture and view network traffic, and a Perl script called `chaosreader` to reassemble network streams into easy-to-read web pages. Services on the network include a simple "stock ticker" program that prints the current value of some imaginary stocks and accepts updates to the stock values. These updates are protected by a nonce, but the transactions can be replayed because of a weakness in the system. During the lab, students are initially given access to the host `eve`, but may compromise the victim hosts `alice` or `bob`, using insertion and rewriting attacks. Finally, students are offered extra credit for reverse engineering the nonce in the stock ticker, allowing forgeries in addition to replay attacks.

This lab demonstrates how easy it is to perform eavesdropping and insertion attacks on typical modern networks. While this knowledge could be used for nefarious purposes, understanding it is essential if students are to

understand the security implications of different network or cryptographic protocol designs.

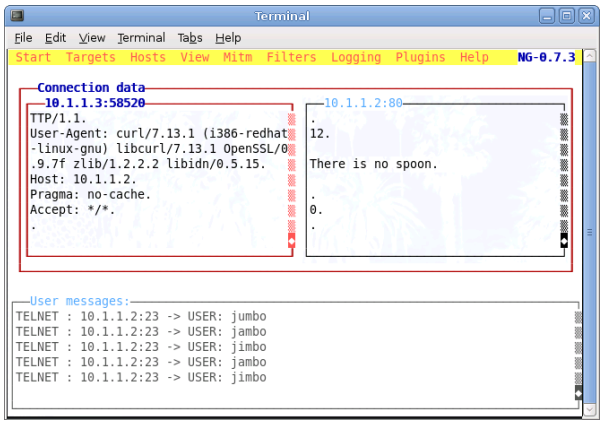

Figure 3: Ettercap eavesdropping on network traffic.

4.6 Network Intrusion Detection Systems

> *You are the system administrator at a small consulting firm, where your server is subject to fairly constant attacks from the Internet. Most of these attacks are automated, but some recent attacks seem like they might have a live human on the other end. Your boss has suddenly become very concerned about the nature and quality of these attacks. He asks you to set up a network intrusion detection system, let it run for a while, and draft a report detailing what the NIDS illuminates.*

Network intrusion detection systems (NIDS) monitor networks for traffic that matches a set of rules which are defined in a simple language. Adjusting, creating, or eliminating rules is similar to writing `iptables` rules. Packets which match a rule are logged in a database for later analysis, typically performed with a web frontend allowing filtering and exploration of the data.

The major cost of operating a NIDS is the maintenance tasks of tuning, authoring, and deleting rules, as well as investigating alerts. Rules which would indicate a serious problem in one environment may be meaningless in another. In this lab, students are responsible for analyzing the significance of a previously collected set of alerts, and for recommending whether the rules generating the alerts should be kept, eliminated, or modified, based on a description of the local network. Students create two new NIDS rules from scratch, and also interpret three network traces captured with `tcpdump`.

The two main pieces of software used for this lab are the Snort NIDS[13] and the BASE (Basic Analysis and Security Engine) frontend.[14] The database of alerts used was created over the course of several weeks using a real production server (with permission). After collecting the alerts, the data was anonymized by obscuring host names and IP addresses, changing usernames, and removing other sensitive information.

This lab is the least dynamic of the set. While it utilizes a complex and powerful web application in BASE, the alerts do not arrive while the student works. As a result, it feels more like an interactive pencil-and-paper exercise. Future work on this lab will include more realistic and dynamic arrival of alerts.

5 Technical Discussion

5.1 Experiment Creation

Creating a DETER experiment requires describing a network topology, choosing operating systems for the nodes, and customizing the nodes on first boot. Each experiment's network topology is defined in the ns2 Network Simulator language[15] with DETER-specific extensions, and can be easily authored using a web-based GUI provided by DETER.

When DETER experiments are swapped in, they only contain the data in the chosen disk image. Customizations can be made with an installation script (defined in the topology) which runs upon first boot. DETER provides a facility for creating your own disk images, but this seemed like overkill since each lab only required a few different pieces of customized software. We used the install script facility to copy all necessary software packages, such as MySQL, lab-specific code, and submission scripts from our `UCLAClass` storage area, and to make any necessary changes to the stock setup (such as adding or removing users). The script is a parent process for a number of smaller installer subprocesses using RPM files, tar balls, or shell scripts. Each subprocess installs one component, and the parent install script will not complete successfully unless each subprocess exits properly. Logins are disabled during the configuration phase by an `/etc/nologin` file, which is removed upon successful completion.

Because the disk images and software archives do not change over time, the installer scripts can be very simple and do not go out of date. This means that the lab software itself requires almost no maintenance. While using older software can be a security risk, this is not a major concern because DETER nodes are firewalled from the outside world.

5.2 Turning a Testbed Into Courseware

Ensuring that the labs on DETER are secure was a major concern. Our course is not collaborative, so we did not

want students to be able to intentionally or accidentally share their work. Avoiding this was a challenge because DETER and Emulab were originally designed to foster collaboration — not privacy — between group members. This forced us to bend the user model of DETER a bit in order to meet our needs.

A single DETER experiment can only be swapped *in* or *out* — one experiment cannot be swapped in by multiple users at the same time. Additionally, home directories are world readable by default, and are mounted via NFS on any experiment to which a user has access. The problem is that students need root privileges for our labs, which can be used to modify other users' files via NFS mounts. In order to solve these problems and provide private storage for each student, we currently isolate each student by assigning them to their own dedicated subgroup containing a complete set of experiments. Recently, however, the DETER developers have implemented a "classroom project" setting that provides equivalent features without dedicated subgroups and provides more protection to TA and instructor accounts.

Unfortunately, there is still a problem that we hope eventually to solve. The NFS security model makes it impossible (as far as we can tell) to both keep the installation media for the labs private *and* give the students root access to the experimental nodes. This is because the root user requires this access to perform the initial installation. We can unmount the NFS volumes after the lab is swapped in, but this is "security by obscurity" — the volumes can be easily re-mounted by a knowledgable user with root access. This is not a problem for projects which do not rely on hidden knowledge in the lab environment. However, if DETER becomes popular for coursework, this vulnerability is sure to become widely known.

5.3 Student Support

Supporting an online course can be challenging. While lectures and lab manuals can easily be downloaded at home, there is no substitute for live, interactive support. For simple course management tasks such as announcements and a forum, we used UCLA's CourseWeb.[16] For student interaction, the online MS at UCLA has videoconferencing hardware at its disposal. However, due to the nature of the labs and the schedules of the students, we chose to use old-fashioned support solutions: email, instant messaging and GNU `screen`.[17]

Email is the most popular support tool by far. Since many of our students work full time, it is much easier for them to send questions to the TA as they work through the labs at their own pace. The TA can reply when convenient, sending class-wide email, updating the class forum, or making fixes to the lab software when necessary.

For supporting office hours, we use instant messaging and GNU screen. GNU `screen` is a powerful terminal multiplexer with many features. For the purposes of our class, `screen` is used primarily because it allows multiple users to connect to the same terminal session. During office hours, the TA logs in to an instant messaging network chosen by the students, and can swap in a special version of the current experiment in the `discussion` group if necessary. After swapping in the experiment, the TA can log in and start a `screen` session to share with any students requiring interactive assistance.

This allows the TA to chat with students, answer questions, and share a terminal with one or more students to support them in real time — with nothing more than an ssh client and the testbed. `screen` allows both users to enter input, so the TA can choose to give direct assistance or merely observe the student and answer questions. Additionally, since the TA is a member of all individual student groups, the TA can also directly offer assistance in the students' experiments.

6 Lessons Learned

The labs have been a great success in the six course offerings since they were developed. Students enjoy work that is relevant to their lives, and our students in particular repeatedly express appreciation for the fact that the labs use real software in realistic environments. This is especially important for computer security because of its unique intersection of theory and practice. We have even received access requests from graduate students who were not enrolled in the class!

In general, students said that they were able to complete the labs in what we felt was a reasonable amount of time. While students with more experience were often able to complete the work faster or develop more creative solutions, students with less experience still scored comparably. Additionally, DETER seemed fairly transparent as a platform; there were very few DETER-specific questions during the course of the class.

One indication of the immersive quality of the labs came from the Forensics lab. Students sometimes find "evidence" in the Forensic images that is an artifact from the lab creation process. For example, because the disk images were installed, attacked, and archived within the span of a few days, students may find evidence of the root user installing new software. Some students notice the incongruous fact that no timestamps on the machine are older than a few days, even though the scenario implies otherwise. However, because they believe in the realism of the labs, students either note this as a sidebar in their report, or more commonly, incorporate the artifacts into their analysis. One welcome, but unexpected, outcome of the Forensics lab is the recognition that multiple reasonable conclusions can be drawn from the same

evidence — a truth with meaningful consequences under the law. This has resulted in extremely creative analyses and also illustrates the burden of detail inherent in creating open-ended, "realistic" labs.

More generally speaking, the open-ended and non-linear nature of the labs has had unexpected, but mostly positive, consequences. In many scenarios, if students know what to look for, they can "solve" the puzzles without much investigation. Yet, other students may spend hours working and still fail to draw correct conclusions. However, we think this is a feature, not a bug, because we want to emphasize analysis, research, careful reading of the material, and critical thinking rather than exhaustive search. In order to facilitate this process, we are relatively forthcoming with hints and advice, serving as a guide to the students rather than a proctor, and continually tweak the lab manual in order to make the assignments and instructions more clear.

7 Future Work

There are many different subjects we could cover through similar labs. For example, a lab on anti-spam, or anti-malware techniques could be interesting, as would distributed denial-of-service defense or botnets. We could also explore Red Team-style exercises on DETER in conjunction with new or old topics, using long-running experiments as "arenas" for student attacks and defenses. Unfortunately, our 10-week quarter does not have room for additional material. One potential solution is to break these labs up into smaller pieces, and assign a different selection of homework during each offering.

We are especially interested in making the exercises less static. Realism is one of our guiding principles, which in turn motivates extremely detailed environments. Unfortunately, it is hard to simulate or synthesize those details, and, in the case of the Forensics lab, it is difficult even to fix an error without introducing a new one. This means that although we cannot, in good conscience, give out answer keys for fear that they will end up online, withholding information may reduce student learning. Our compromise has been to give thorough feedback during grading and invite students to discuss any questions they might have on an individual basis.

In response to the general difficulty of content creation, we have started working on tools to make the lab creation process more flexible and dynamic. Specifically, we have a prototype system for generating forensic images which utilizes Expect,[18] an automation tool for operating systems. Our tool allows us to script actions which create the forensic image on a real system. These actions can include installing software, impersonating users, generating network traffic, and the like. The main advantages of such a tool are control (the tool can execute much more complex and precise behaviors than one author could) and repeatability (fixing a bug in a forensic image is simplified to fixing a bug in the script and re-creating the image). Work on this tool is ongoing.

8 Conclusion

The labs are a great success, meeting and exceeding our expectations. The labs compare very favorably to "offline" projects, and they are easy to use. We continue to find that students enjoy the immersion, imaginative scenarios, and experience with real software systems. We also heartily recommend DETER as a platform for educational software. It has been an excellent and powerful tool; we could not have easily created these labs without it. DETER's administrators have worked hard to support education; we and others appreciate this and will continue to rely on DETER for this purpose [2] in the future.

References

[1] BENZEL, T., BRADEN, B., FABER, T., MIRKOVIC, J., SCHWAB, S., SOLLINS, K., AND WROCLAWSKI, J. Current Developments in DETER Cybersecurity Testbed Technology. In *CATCH 2009* (2009).

[2] DETER TESTBED. Education on DETER. `http://www.isi.edu/deter/education/`. Accessed Friday, May 28, 2010.

[3] WHALEN, S., ENGLE, S., AND ROMEO, D. An Introduction To Arp Spoofing. `http://www.leetupload.com/database/Misc/Papers/arp_spoofing_slides.pdf`, 2001.

[4] WHITE, B., LEPREAU, J., STOLLER, L., RICCI, R., GURUPRASAD, S., NEWBOLD, M., HIBLER, M., BARB, C., AND JOGLEKAR, A. An integrated experimental environment for distributed systems and networks. In *OSDI 02* (Boston, MA, Dec. 2002), USENIX, pp. 255–270.

Notes

[1] http://qemu.org/
[2] http://www.virtualbox.org/
[3] http://www.vmware.com/
[4] http://www.deterlab.net/
[5] http://moinmo.in/
[6] http://lasr.cs.ucla.edu/seclabs/
[7] http://nmap.org/
[8] http://netcat.sourceforge.net/
[9] http://www.chkrootkit.org/
[10] http://www.openwall.com/john/
[11] http://e2undel.sourceforge.net/
[12] http://ettercap.sourceforge.net/
[13] http://www.snort.org/
[14] http://base.secureideas.net/
[15] http://www.isi.edu/nsnam/ns/
[16] http://courseweb.seas.ucla.edu/
[17] http://www.gnu.org/software/screen/
[18] http://expect.nist.gov/