Class Capture-the-Flag Exercises*

Jelena Mirkovic
USC/ISI

Peter A. H. Peterson
USC/ISI

Abstract

The field of cybersecurity is adversarial – the real challenge lies in outsmarting motivated and knowledgeable human attackers. Sadly, this aspect is missing from current cybersecurity classes, which are often taught through lectures and occasionally through "get your feet wet" practical exercises. We propose Class Capture-the-Flag exercises (CCTFs) to revitalize cybersecurity education. These are small-scoped competitions that pit teams of students against each other in realistic attack-defense scenarios. We describe how to design these exercises to be easy for teachers to conduct and grade, easy for students to prepare for and a lot of fun for everyone involved. We also provide descriptions of CCTFs we have developed and recount our experiences of using them in class.

1 Introduction

Cybersecurity is a unique field of science and engineering, because its main challenges are not solely dictated by technological limitations or the theoretical complexity of the underlying problems. Rather, cybersecurity advances are driven by the clash of minds – researchers create new defenses and criminals adapt their attacks in response. This adversarial game is at the heart of each cybersecurity challenge, but it is sadly absent from cybersecurity education.

The ACM defines three types of learning outcomes in their Computer Science Curricula 2013 document [4]:

1. **Familiarity** – A student understands the concept at the theoretical level. This is usually achieved via textbooks and lectures.
2. **Usage** – A student understands the concept and can apply it correctly when a situation requires it. This is usually achieved by a mix of lectures and practical exercises that are well-specified and demonstrate basic concepts.
3. **Assessment** – A student understands the concept, can correctly recognize the given concept in practice, can weigh it and related concepts as solutions to some problem and can apply each of them correctly. This is usually achieved by a mix of lectures and practical exercises that are open and allow for exploration and decision-making.

As students advance on their learning path from *familiarity* to *assessment*, student engagement, interest and retention increase. Currently, many security classes are taught the old-fashioned way, using textbooks and lectures, with focus on theory and case studies. This leads only to *familiarity* learning outcomes and results in narrowly educated and poorly trained professionals. It also reduces retention in these fields [6] and in Computer Science in general, because active student participation increases their motivation. Commendably, some classes include hands-on exercises to demonstrate concepts taught in lectures [2, 8, 3] and lead to *usage* learning outcomes. This is necessary but not sufficient. Students acquire some practical skills performing these exercises but do not get to experience the adversarial nature of the field, nor do they get to apply their newly acquired skills to novel situations where success depends on their inventiveness, ability to make the right decisions quickly and work in a team. These skills are needed daily in a cybersecurity career. Accordingly, the better equipped our students are with these skills when they graduate, the more quickly and competently they will enter the work force.

We propose to revitalize security education through class capture-the-flag exercises (CCTFs). These are small-scoped Capture-the-Flag (CTF) exercises, designed to require a few weeks of preparation from stu-

*This work is in part supported by the DHS grant N66001-10-C-2018. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Homeland Security.

dents and to be conducted in as little time as a two-hour class. They engage teams of students in attack-defense scenarios. Each team plays both the defense and the attack role, which enables them to understand and acquire the adversarial thinking model needed for a cybersecurity career. We have designed CCTFs to require minimal support from teachers – their setup and scoring is automated and they are conducted on the DeterLab testbed [9], an open testbed for cybersecurity experimentation with mechanisms that ensure isolation of resources and activities among teams. Each CCTF focuses on one security topic (e.g., cryptography, exploits, denial-of-service, etc.). This scoping enables students to exercise skills they have recently learned in class and possibly practiced through hands-on exercises. After each CCTF, a teacher leads an in-class post-mortem analysis on the event, enabling students to identify what they did right or wrong and to improve for future competitions. We believe that these exercises will help students achieve the *assessment* learning outcome and own the material they learned in class at a deeper, more masterful level – while also having a lot of fun.

In this paper we describe ten CCTFs; four of which we have already developed and six that are under development. We also recount our experiences of using CCTFs in an undergraduate security class at USC for two consecutive years. We further detail our plans for the public release of these materials through the DeterLab education web site [2] in Fall 2014.

2 DeterLab

The DeterLab testbed [9] is the deployment platform for our exercises. DeterLab is an open and free-for-use research and education testbed funded by the Department of Homeland Security and the National Science Foundation. Co-hosted by USC/ISI and UC Berkeley, it currently includes approximately 500 physical machines and is used by more than 3,700 users in 30 countries. About 75% of these are educational users.

DeterLab is based on Emulab technology [12], which enables flexible, automated setup and remote experimentation. When an environment is instantiated on DeterLab, users gain exclusive, root access to a set of physical machines for a limited time. The machines run operating systems and applications of the user's choice and are organized into a user-specified topology. Traffic between machines in the experiment is fully contained and does not affect other experiments on the testbed, nor can it get out to the Internet. Machines within DeterLab experiments can only be accessed via an SSH tunnel that crosses the well-policed gateway machine. This makes the DeterLab testbed an ideal platform for adversarial exercises because all the potentially harmful traffic and actions are fully contained and thus harmless to the general public.

Interaction with DeterLab occurs primarily through a web interface with graphical tools for creating and managing experiments. This makes the testbed accessible from anywhere, enabling students and teachers to interact from different physical locations (e.g., a dorm room, a home, an office, another state or country, etc.). Once the experiment receives physical resources, users can log on to them using SSH and set up any traffic generation and monitoring they choose. The DeterLab testbed also has many traffic generation, visualization and experiment monitoring tools that facilitate easy experimentation for novice users.

Like our other publicly available educational materials (e.g., hands-on exercises [7]), all CCTF exercises will be hosted at DeterLab's education site [2]. Anyone will be able to contribute new exercises to our repository. We will curate each such submission to enforce a common format and to ensure that they all achieve our desired level of quality and ease of use.

3 Class Capture the Flag Exercises

Capture-the-flag exercises are a great motivator for learning about security, because they pit teams against each other, challenging them to outwit an opponent using their security knowledge and skills. This environment provides the experience of fighting a live enemy on a time budget, incentivizes participation and builds team spirit. It also leads to mastery of the course material through the process of decision making, synthesis of knowledge and application to real problems.

3.1 CCTFs vs CTFs

Many Capture-the-Flag (CTF) exercises are organized annually in the US in academic, industry and government circles. These include the International CTF [11] organized by UC Santa Barbara, National Collegiate Cyber Defense Competition (NCCDC) [1], DEFCON [5] and many others. While large-scale CTF events are valuable in many ways, our CCTFs have distinct properties making them uniquely aligned for the educational setting. Table 1 and the following paragraphs summarize the differences between traditional CTFs and our CCTFs.

1. **Lightweight** (F1, F2 and F9 from Table 1) – Most CTF exercises require at least one whole day of activities and many weeks of dedicated preparation. In contrast, CCTFs have limited scenarios that require two to three weeks of preparation, one to two hours of time investment per day, and about two hours to conduct the exercise. This lowers the burden on

Feature		CTFs	CCTFs
F1	Preparation	A few months	A few weeks
F2	Duration	1-2 days	2 hours
F3	Topic	Hacking	Exploits, DDoS, DNS, BGP, crypto, etc.
F4	Team roles	Blue or Red, rarely both	Both Blue and Red
F5	Team pairing	All against one or all against all	Playoff
F6	Occurrence	Annual or semi-annual	2-3 times per semester
F7	Postmortem	Rarely	Always
F8	Advancement paths	None	Built-in
F9	Difficulty	Expert	Beginner to intermediate

Table 1: Comparison of CTFs and CCTFs

teachers and barrier to entry for students, making it possible to advance their security knowledge while also tending to their other classes and time obligations such as work, exams, extracurricular activities, etc.

2. **Frequent and with a post-mortem** (F6, F7 and F8 from Table 1) – Most CTF exercises occur 1-2 times per year and have a few winners and many losers. While everyone learns a lot from participating, being on a losing side may discourage students from the security field. This is especially true for disadvantaged students lacking in educational resources, background knowledge or skills (e.g., students with little personal experience or from schools that may not have a strong security emphasis). These students can be at a distinct disadvantage and may even enter the competition feeling inferior to students from larger institutions.

   Our CCTFs can occur multiple times per semester. Their short duration and minimal investment in preparation make it possible for students to compete multiple times to iteratively hone their skills throughout the semester. Each CCTF is followed by post-mortem discussions facilitated by course instructors, where the teams discuss their actions, tactics and observed effects. In other words, losing teams can learn from their mistakes and apply that knowledge in future CCTFs.

3. **Two-sided** (F4 from Table 1) – Some CTF exercises are defense-only and use a professional offense team, provided by NSA or industry volunteers. Other exercises are offense-only, where teams need to find and exploit vulnerabilities in a given system. CCTFs are two-sided with student teams providing both the offense and the defense, or may even be three-sided with a White team providing legitimate user traffic on a victim network. While we provide scenarios, rules of engagement, necessary software and automated set up, the execution of exercises and the sophistication of attacks and defenses come from students. This teaches students basic defense and offense skills for secure system design, implementation and testing, while limiting the burden on teachers adopting the exercises and on the DeterLab testbed facility supporting them.

4. **Playoff-based** (F5 from Table 1)– Most CTF exercises have all teams compete against each other (or a common enemy). CCTFs are structured as playoff competitions, with pairs of teams competing against each other in multiple rounds. This ensures that each team has multiple chances to win over the course of a semester, as they fight teams with varied skills. Also, making each exercise a one-on-one competition provides valuable focus given the short duration of CCTFs.

5. **Versatile** (F3 from Table 1) – Most CTF exercises focus on performing exploits or protecting vulnerable code from being exploited. However, the general design of our CCTFs enable discrete exploration of a wide range of security scenarios. For example, scenarios can focus on poor protocol design, distributed implementation (where some elements of the scenario are unknown or are outside of students' control), resource limitations and so on. CCTFs focused on specific security problems let students exercise knowledge obtained at various points in a semester. At the same time, multiple issues can be combined in one CCTF, helping teach them to consider security holistically and to evaluate the trade-offs of every decision, just as they would in careers as security professionals.

3.2 Ethical Issues

We understand that teaching students how to attack systems raises important ethical issues and may be contro-

versial. However, we feel that the critical benefits of doing so outweigh the drawbacks. Specifically, we feel that future security professionals must be exposed to intelligent and adversarial behavior because it is one of the key challenges in the world of real systems and networks. Accordingly, we cannot expect students to master the secure design and defense of systems unless they can practice doing so in an adversarial environment.

As an analogy, we do not expect doctors to learn how to treat the sick by merely reading about diseases. Instead, doctors undergo extensive training where they examine and treat patients – first under supervision and then with increasing independence. It is not enough to simply read about diseases because: (1) disease presentation varies considerably and many diseases have similar symptoms, (2) the same disease may progress differently depending on the patient and the treatment and (3) doctors need extensive practice in diagnosis and treatment before they can perform these skills quickly and accurately. We argue that security students need similar exposure to attacks so they can learn to defend against them. Security attacks evolve continuously, there are many variations, many attacks look similar and they may progress differently in different networks. Only repeated exposure to these can train students to anticipate them, detect them quickly and accurately select and apply the right defense.

If it is critical that students be exposed to attacks so that they may learn "in the trenches," then we need a source for safe but effective exposure. Doctors have an unending supply of people requiring medical attention. However, while cyberspace is full of beleaguered networks, we obviously cannot give students production-level systems to defend. One approach might be to recreate attacks in testbeds based on historical data. Unfortunately, there is – for many reasons – very little publicly available detailed data about real attacks. Even if such data were available, "historical reenactments" of attacks could require a large effort to engineer, would be tightly coupled to (and limited by) the data and would have static solutions because they would lack intelligent attackers capable of adjusting their strategy based on the defenders' actions. Hiring a dedicated offensive team makes little sense for classes because they would be very expensive and would easily defeat most students. Having the teacher play the attacker would be similarly unfair, not to mention extremely burdensome for the teacher.

Our choice of having students play both the attackers and the defenders has several nice properties: (1) it lets students pit their wits against adversaries likely to have similar security skills, which helps equalize the strengths of attackers and defenders, (2) in the process of performing the exercises, students acquire skills essential for security-related careers where an adversarial mindset is necessary for anticipating future offensive actions (e.g., penetration testing and system administration) (3) it provides deeper experience and spices up the exercises (e.g., offense is always more fun than defense), (4) students may be more motivated to participate when they fight their peers than an unknown third party and (5) using students on both sides keeps exercises fresh and limits many types of costs.

We have implemented the following safeguards to ensure that students do not misuse their offensive skills.

1. **Required ethical offense training** – We have produced a set of slides on ethical offense and require each teacher that uses our CCTFs to cover them in class before the exercises. The materials discuss ethical behavior during the exercises, the laws that apply to use of offensive skills in real networks, and the consequences one may incur for such behavior. Before we publicly release CCTFs we will also develop a quiz about ethical offense that each student must pass in order to participate in CCTFs.

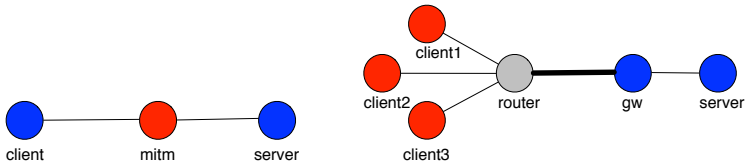
2. **Exchanging the attacker and defender roles** – Each student is equally exposed to roles of attacker and defender as each team plays both the defender and the attacker role.

3. **Post-mortem analysis** – After each CCTF exercise instructors perform post-mortem analysis with the attack and the defense team. This reinforces in students the understanding that the whole experience serves to improve their learning of the material as opposed to teaching them how to attack systems. Both attackers and defenders analyze their strategies and together seek ways to improve them for future competitions.

Finally, we would like to emphasize that our CCTFs do not teach students any offensive techniques that are not already widely known and available on the Internet. In fact, students learn most of the offensive skills from the well-known web pages (assembled by the security community) that describe the techniques in question.

3.3 Seed CCTFs

To date, we have developed four CCTF exercises, three of which have been used in an undergraduate Introduction to Security class at USC. We also have plans to develop six more exercises. Below we detail all the planned and developed exercises. We refer to the defense team as the Blue team and the attackers as the Red team. Some exercises also engage a White team that provides services, such as setting up routing or generating legitimate traffic. For these exercises each team plays the role of



(a) Secure Data Topology (b) Resilient/Protected/Secure Server Topology

Figure 1: Topologies

a Blue team for their own resources, a Red team for another experiment and a White team for the third experiment.

Secure Data (developed and used) – Experimental topology for this exercise is given in Figure 1(a). Blue team has access to the machines `client` and `server` and the Red team has access to machine `mitm`.

The Blue team designs and implements several simple cryptographic approaches: a monoalphabetic cipher, a polyalphabetic cipher, a polygram cipher and a homophonic cipher. They also build server and client software that exchange information using these ciphers. The encryption key is stored on the machines prior to the exercise and cannot be changed manually during the exercise (but can be changed programmatically). At the time of the exercise the teacher provides each team with the set of messages to be transmitted between nodes. The teacher also informs both teams which one of the four ciphers must be used during communication. The Red team must capture messages exchanged by the Blue team between `client` and `server`, break the encryption and read them. The Red team obtains one point for each deciphered message and the Blue team obtains one point for each message that the Red team did not decipher.

Resilient Server (developed and used) – The experimental topology for this exercise is given in Figure 1(b). The Blue team has access to `gw` and `server`, while the Red team has access to `c1`, `c2` and `c3`. Only the teacher has access to `router`. The bandwidth on the link between `router` and `gw` is ten times higher than bandwidth of other links in the topology. The Blue team sets up a simple web server that serves ten static web pages with predefined names. They also must develop traffic monitoring software that enables them to quickly detect denial-of-service attacks and to pinpoint any characteristics they may use for filtering. Finally, the Blue team sets up filtering at `gw` as needed to protect the server. The Red team chooses one of the machines `c1` to `c3` to be the legitimate client machine, while the other two will be attackers. This choice is only known to the Red team. Approximately every 10 seconds, the legitimate client machine must send one request to the server, asking for a web page. The two attack machines can, at will, choose to mimic the legitimate client's timing and behavior or they may be used to attack. The goal of the Red team is to overwhelm the server machine with denial-of-service. The Red team obtains one point for each legitimate client's request that is processed with greater than 0.5s latency (or not processed at all) and the Blue team obtains one point for each legitimate client's request that is processed with less than 0.5s latency.

Protected Server (developed) – Experimental topology and access controls for this exercise are same as for *Resilient Server*. The Blue team sets up a LAMP server and writes code to mimic an online bank where a user can register, deposit some money, withdraw it and check their balance (required APIs for the bank software are defined in the exercise). The Red team controls machines `c1` through `c3` and uses one of them as a legitimate machine (this choice is only known to the Red team) and the other two as either legitimate or attack clients. The Red team attempts to exploit the server through SQL injection or other vulnerabilities to create unexpected behavior (denial of service attacks are not allowed). The server must log each request and the state of the DB after processing each request. The Red team obtains one point for each request that leads to an unexpected server behavior (e.g., withdrawing money that the user did not deposit, checking a balance of a different user, etc.) and the Blue team obtains one point for each well-processed request (including malformed requests that they have detected and refused to process).

Secure Server (developed and used) – Experimental topology and access controls for this exercise are same as for *Resilient Server*. The Blue team receives a LAMP server with buggy PHP code that implements an online bank. They must patch this code to secure it against various attacks and develop monitoring and filtering software to detect and filter DDoS attacks. The Red team controls machines `c1` through `c3` and uses one of them as a legitimate machine (this choice is only known to the Red team) and the other two as either legitimate or attack clients. The Red team attempts to exploit the server, either through SQL injection or through another vulner-

ability, to create unexpected behavior. They can also launch denial-of-service attacks. The Red team obtains one point for each request that leads to an unexpected server behavior or for each legitimate client's request that is processed with more than 0.5 s latency. The Blue team obtains one point for each well-processed request (including malformed requests that they have detected and refused to process). Well-processed requests must be handled with less than 0.5 s latency.

Secure DNS Infrastructure (planned) – The Blue team designs and implements a way to securely disseminate DNS information for their web server and prevent DNS hijacking. Since not all clients may use secure DNS the Blue team also devises ways to monitor for, detect and respond to DNS hijacking. The White team in the exercise provides legitimate customer traffic – some customers run secure DNS queries while others are legacy customers. The White team also sets up an emulated DNS hierarchy (root and .com servers) which are off limits to the attackers. The Red team's goal is to hijack the Blue team's traffic by performing DNS cache poisoning or ARP poisoning in the Blue team's network. They have no knowledge of the Blue team's network architecture or the DNS server's code.

Secure Routing Infrastructure (planned) – The Blue team designs and implements a way to securely disseminate BGP routing information for their web server, while preventing BGP prefix hijacking. The White team provides legitimate customer traffic and will also set up a network that transits traffic between the Red team and the Blue team (the legitimate customers). Some, but not all, routers in this network will run Secure BGP. Directly attacking this network will be off limits to the Red team. The Red team's goal is to hijack the Blue team's traffic by performing BGP prefix hijacking or ARP poisoning. They have no knowledge of the Blue team's network architecture or their BGP settings prior to the exercise.

Attack Forensics (planned) – The Blue team provides a web server, running a set of logging and monitoring services in unsupervised fashion. The White team provides legitimate clients' traffic for the server. The Red team breaks into the server machine and performs any attacks they desire. The Blue team's goal is to determine when the attack occurred, what happened and to collect evidence to prove their findings.

Polymorphic Worm (planned) – The Blue team sets up a network with several vulnerable services that are disclosed to the Red team. The White team uses these services, providing legitimate traffic. The Red team writes a polymorphic worm that exploits one or more vulnerable services to spread in the network. The Blue team's goal is to detect this spread and stop it before all their machines are compromised.

Privacy-Safe Data Release (planned) – The Blue team sets up a network with several services and the White team provides legitimate traffic. The Blue team collects network traces and releases them "to the public" in an anonymized form. The Red team attempts to deanonymize these traces and identify the activities of any legitimate client. The Red team may interact with the Blue team's servers in this process.

Botnet Identification (planned) – The Blue team sets up a network with several vulnerable services that will be disclosed to the Red team. The White team provides legitimate traffic in addition to an emulated e-mail server. The Red team exploits as many machines as they can, organizes them into a botnet and uses them to send spam to the e-mail server. The goal of the Blue team is to identify which machines are bots and to interpret their command and control traffic.

3.4 Running CCTFs

All CCTFs can be used as intensive and short-duration one-off exercises in which the Blue team develops new defenses and the Red team develops new attacks. Some CCTFs may be better suited to become ongoing duels running over several weeks where the Blue team tries to maximize the time when their goals are met, while the Red team attacks asynchronously. This would enable students to evolve their skills over time, leading to more challenging scenarios. To maximize student involvement, these ongoing duels could be scheduled for times when school is not actively in session, such as spring, summer or winter breaks.

A teacher would start by running CCTFs within their class, pitting teams of students in the same class against each other. As teachers become more comfortable with the CCTFs and as CCTFs see wider adoption, we envision that teams from different schools could schedule competitions against each other. This would require tight synchronization between teachers of both classes, but should spice up the exercises and increase competitiveness. We expect to offer scheduling of such inter-school CCTFs on DeterLab starting in Fall 2015.

Scoring CTF exercises presents a challenge [10] in any setting, because teams may attempt to cheat in order to elevate their scores. Some CCTF exercises require teams to capture flags that are cryptographic keys or hidden data, which makes scoring easy. Other exercises require teams to maintain the availability and integrity of services and data, and must be scored by probing or by logging of traffic and data. We have developed custom scoring mechanisms for the four existing CCTFs and expect that the remaining CCTFs will require a similar effort.

For example, scoring in the Secure Data exercise

is performed by logging and comparing decrypted and cracked messages. Each message has a sequence number appended before encryption so it can be easily verified in logs. Messages are supplied by the teacher at the start of CCTF, which minimizes the risk of leakage. As another example, scoring for the Resilient Server exercise is based on traffic captured by a secure machine. Post-processing on the captured data determines the service quality to the legitimate client and other scored metrics.

4 Experiences and Conclusions

We have used the Secure Data and Secure Server CCTFs in our 2013 offering of the Computer Science undergraduate "Introduction to Security" class at USC, and Secure Data and Resilient Server CCTFs in our 2014 offering. In 2013, there were 10 students enrolled in the class, which amounted to three groups of three to four students each. In 2014, there were 25 students divided into five groups of five students each. Each year, team assignments were done manually by the teacher. We surveyed the students at the beginning of the class about their skills and background knowledge, and used this information to create balanced teams. We kept the team composition the same for both CCTFs in the same class. While student skills different widely we did not notice any dominant behaviors by more skilled team members. We allowed teams to divide tasks among themselves as they thought best. While there was imbalance in this division (more skilled members took on more complex tasks), each member made significant contributions to team effort. Each exercise had 2–3 weeks of preparation and ran for 2 hours. We planned the curriculum so that lectures covering the material needed for each CCTF, and hands-on exercises demonstrating key concepts, were completed prior to the preparation time. Our course can thus be viewed as a combination of two parts, with each part covering several topics. Each topic is first covered in lecture, then it is demonstrated via a hands-on exercise assigned as a homework, and finally it is mastered by students in a CCTF.

In 2013 we did not provide any milestones for the teams and did not check their progress. After observing that most teams start development late and do not integrate the code on time, we introduced a set of milestones with suggested completion dates in 2014. This helped streamline development and integration. It also helped balance the development of defensive and offensive code. In 2013, in the Secure Data exercise all teams completed defensive tasks (encryption development) but none automated the offense (encryption cracking). The situation was the opposite in the Secure Server exercise where all teams focused on offense and did not sufficiently harden nor monitor their server. After the introduction of milestones in 2014 both offensive and defensive tasks were completed by all teams.

The Secure Data exercise was challenging to students in both years. We believe that this is mostly because it was their first experience with CCTF exercises. Teams had problems with achieving milestones on time, with network-level tasks such as message interception using `iptables` and `netfilter`, and with packing and unpacking of binary messages. Blue team tasks were more easily completed than Red team tasks, where automating the cracking of encryption was the most difficult. In 2013, no team completed this task. In 2014, all teams completed automated cracking but it was not effective. During the exercise two teams could decipher parts of messages in automated fashion but did not know how to improve their code to obtain the entire key. To allow time for teams to fully test their code (and not just its parts), the teacher should require integration at least a few days to a week before the actual exercise.

Secure Server and Resilient Server exercises were well-enjoyed by students and tasks were more easily completed as students were familiar with CCTF preparation. In 2013, all teams focused more on offense than on defense. In 2014, all teams completed both the defensive and the offensive tasks. While team sophistication and success in the exercise differed, all teams managed to conduct at least one effective attack and to defend against at least one attack from the Red team. We also observed that some teams used offensive code found on the Internet (which they were allowed to do), but did not quite understand how to use it. This led to some testbed problems that we were able to quickly diagnose and solve during the exercise.

Overall, we observed that these exercises spiked student interest in learning and let them master the skills they acquired through lecture and hands-on exercises, as well as practice teamwork. Students were graded on their contributions (the tasks they completed) to the team. We deliberately did not grade teams based on performance, because we wanted to nurture a collaborative learning environment and to allow students to experiment and make mistakes. This was very successful as we observed teams helping each other during the exercise, with very little competition. We hope that conducting CCTFs between schools in the future should increase competitive aspects of the exercises, while post-mortems conducted jointly by teachers should still nurture the collaboration.

Finally, we observed that beyond security topics these exercises represent a nice way to teach students how to use scripting languages, such as Python or Bash, how to use network monitoring tools like `tcpdump`, write network code, test their code and how to administer Linux systems. We believe that these skills will be very useful to them in their future careers.

References

[1] National Collegiate Cyber Defense Competition. http://www.nationalccdc.org/.

[2] DETER Project. DeterLab Education Web Site. http://education.deterlab.net.

[3] Wenliang Du. SEED: Developing Instructional Laboratories for Computer SEcurity EDucation. http://www.cis.syr.edu/~wedu/seed/index.html.

[4] ACM/IEEE-CS Joint Task Force. Computer Science Curricula 2013. http://cs2013.org.

[5] DEF CON Communications Inc. DEFCON. http://www.defcon.org.

[6] Paul H. Kvam. The Effect of Active Learning Methods on Student Retention in Engineering Statistics. *The American Statistician*, 54(2):136–140, 2000.

[7] J. Mirkovic and T. Benzel. Teaching Cybersecurity with DeterLab. *EEE Security and Privacy Magazine*, 10(1):73–76, January/February 2012.

[8] SeattleTestbed. Seattle in the Classroom. https://seattle.cs.washington.edu/html/education.html.

[9] USC/ISI and US Berkeley. DeterLab testbed. http://www.isi.edu/deter.

[10] G. Vigna. Teaching hands-on network security: Testbeds and live exercises.

[11] G. Vigna. The UCSB iCTF. http://ictf.cs.ucsb.edu/index.php.

[12] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *OSDI*, 2002.