Experiences In Cyber Security Education: The MIT Lincoln Laboratory Capture-the-Flag Exercise [*]

Joseph Werther, Michael Zhivich, Tim Leek
MIT Lincoln Laboratory
POC: joseph.werther@ll.mit.edu

Nickolai Zeldovich
MIT CSAIL

ABSTRACT

Many popular and well-established cyber security Capture the Flag (CTF) exercises are held each year in a variety of settings, including universities and semi-professional security conferences. CTF formats also vary greatly, ranging from linear puzzle-like challenges to team-based offensive and defensive free-for-all hacking competitions. While these events are exciting and important as contests of skill, they offer limited educational opportunities. In particular, since participation requires considerable *a priori* domain knowledge and practical computer security expertise, the majority of typical computer science students are excluded from taking part in these events. Our goal in designing and running the MIT/LL CTF was to make the experience accessible to a wider community by providing an environment that would not only test and challenge the computer security skills of the participants, but also educate and prepare those without an extensive prior expertise. This paper describes our experience in designing, organizing, and running an education-focused CTF, and discusses our teaching methods, game design, scoring measures, logged data, and lessons learned.

1 INTRODUCTION

In April of 2011, MIT Lincoln Laboratory organized a CTF competition on MIT campus to promote interest in and educate students about practical computer security. The competition was structured around defending and attacking a web application server. The target system consisted of a LAMP (Linux, Apache, MySQL, PHP) software stack and WordPress, a popular blogging platform [1]. In order to familiarize participants with the target system and to provide an opportunity to implement substantial solutions, a virtual machine very similar to one used during the competition was made available to the participants over a month before the competition took place. To help participants prepare for the event, we offered evening lectures and labs that discussed defensive and offensive techniques and tools that might be useful during the competition. The competition itself was an eighteen-hour event held over the weekend of April 2-3, during which students worked in teams of three to five to defend their instance of WordPress, while simultaneously attacking those of other teams. A scoring system provided numerical measures of instantaneous and cumulative security, including measures of availability, integrity, confidentiality and offense. Carefully crafted but realistic vulnerabilities were introduced into WordPress at the start of the competition via ten plug-ins authored by the Lincoln team. Since getting a high availability score required a team to run these plug-ins, participants were forced to come to terms with the very real dangers of rapidly deploying untrusted code. WordPress is famous for its extensibility, and plug-in architectures are increasingly common and popular in software engineering. A goal of the MIT/LL CTF was to explore this novel computer security issue.

The event was open to all Boston area students, without pre-requisites or a qualification round, with main motivation including capturing an actual flag (we made a flag that the winning team took home), learning about practical computer security, and taking home a $1,500 first-place prize. Sixty-eight students registered for the event over the course of two weeks in a first-come, first-served basis. Of these, fifty-three actually formed teams, and on the day of the exercise, forty-five showed up in person at 8:30 am on a weekend to compete in the CTF.

Participants' response to the MIT/LL CTF was overwhelmingly positive. After the competition ended, we distributed a survey to ascertain the educational value of this CTF. The survey responses indicate that students learned much about practical computer security both before the competition (in lectures and labs, self-study, and group activities), and during the competition itself (where the time pressures of the competition bring into sharp focus theoretical computer security lessons). While donning the "black hat" to hunt for flaws in code and configurations is certainly fun, we assert that it is also a powerful intellectual tool for challenging assumptions and mindsets. We believe that a CTF is a valuable *pedagogical tool* that can be exploited to engage students in the study of complicated modern computer and network systems. Further, we believe it can be accessible to a much larger

[*] This work is sponsored by OUSD under Air Force contract FA8721-05-C-0002 and by DARPA CRASH program under contract N66001-10-2-4089. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

subset of computer science students than a traditional CTF.

This paper discusses our experiences of designing and running the MIT/LL CTF. We examine the successes and failures, related to both our educational goals and system design, and share our overall conclusions with respect to future improvements. Section 2 discusses our pedagogical approach, and Section 3 describes the educational methods and lectures used for the CTF. Section 4 examines the choices made with respect to selecting the components and techniques used in designing the game and evaluating participants' progress. Section 5 covers the evaluation metrics used to assess teams during the competition. Section 6 analyzes data gathered during and after the event including team scoring trends, reactions to changes in the importance of metrics during the game, and survey results from the participants. Section 7 discusses related CTF exercises and how the MIT/LL CTF is different. Finally, Section 8 offers lessons learned by the MIT/LL CTF staff, and how we plan to change the event in future iterations.

2 Pedagogical Approach

Students learn about computer security in a number of ways. Reading conference papers, journal articles, and books, for instance, allows students to acquire valuable theoretical apparatus and learn what has been tried before. Designing and implementing defensive systems and offensive tools is also valuable, as it requires the application of abstract knowledge to build real working solutions.

In addition to these fairly traditional educational avenues, we believe that practicing defending and attacking real computer systems in real time is also of immense educational value, and that it offers lessons that can't effectively be taught in a classroom. A CTF event is a playground in which students can fail or succeed safely at computer defense, and where it is permissible to engage in attack, without fear of consequences or reprisal. We believe it is crucial for CTF events to include an offensive component, not only because students find it exhilarating, but because it also challenges flawed reasoning and assumptions in tools, techniques, and systems, and leads to a deeper understanding of computer science in general [6].

Despite the significant educational potential of a CTF, many potential participants (i.e., those with a general computer science background or even a few computer security classes under their belt) perceive there to be a high barrier to entry. Unfortunately, they are often right: participating in and learning from a typical CTF competition requires significant skills and background knowledge. For participants with inadequate skills, it can be frustrating and bewildering, as their systems are compromised quickly and repeatedly. They are lucky if they even know what has happened, let alone why.

We are certainly not the first to consider offensive components to be crucial to learning practical computer security. O'Connor et al [11] suggest that framing study of general computer science concepts from a perspective of an adversary encourages student participation and interest. Moreover, they argue that framing the problem in terms of security (e.g., forensics) makes learning about other topics, such as file system formats, much more exciting. George Mason University explored the merits of having students create offensive test cases to exploit the code they developed as part of class assignments [8]. In doing so, they found that the offensive mindset led students to discover vulnerabilities not found through other testing methods. Bratus [2] also argues for adding components of adversarial, *hacker* mindset to the traditional computer science curriculum due to the low-level knowledge it imparts, and its necessity in understanding and effectively implementing secure systems. In designing our CTF, we sought to enable a wider range of participants to benefit from this approach to computer security education.

3 Educational Design

Since our goal was to create not only another CTF exercise, but also a pedagogic tool for teaching computer security, we incorporated several educational components, in the form of lectures and labs, into the MIT/LL CTF. In total, we offered five classes in the month preceding the competition.

The first lecture provided an overview of the CTF game, its mechanics and rules. As part of this description, we presented the game platform and architecture (Linux, x86), as well as the intended target – the WordPress content management system. We also explained and justified the scoring system, with suitable measures of *confidentiality*, *integrity*, *availability* and *offense* (see Section 5). This meeting allowed those who had not taken part in a CTF exercise before to understand the game better and ask questions.

In the second class, we presented the basics of web applications, the WordPress API, and some of the fundamental ways in which its design makes computer security difficult. We did not teach PHP, JavaScript or SQL, even though WordPress makes use of all three, as these details could easily be mastered by the general computer science student in self-study. The intent was not to educate students to the point that they might go off and write a web application; rather, we hoped to orient them in this (perhaps unfamiliar) terrain, providing an overview of the target and sketching the security issues for them to consider on their own.

The third class covered various aspects of Linux server security, also in lecture form. Topics ranged from high-level concepts, including the principle of least privilege, multi-layer defense and attack surface, to low-level discussions of practical details such as firewalls, application configuration, package management and `setuid` binaries. A number of standard tools and packages such as `AppArmor`, `Tripwire`, and `fail2ban` were also explained. Additionally, MIT's volunteer student-run computing group SIPB presented a case study in securing the web application (`scripts.mit.edu`) and virtual machine (`xvm.mit.edu`) hosting services provided to the MIT community.

The fourth lecture discussed web application exploitation techniques including SQL Injection, Cross-Site Scripting, and other server-side and client-side attack vectors. Each topic was addressed from the aspects of vulnerability discovery, exploitation, and mitigation. It was our hope that approaching each issue from all three angles would help the participants build better tools in the weeks leading up to the competition.

The final class consisted of a lab in which the participants were asked to work through computer security challenges with the help of the organizers. For this exercise, we used Google Gruyere [7]. This site allowed each student to stand up a separate instance of the exercises and practice finding and exploiting the plethora of issues discussed in the previous lecture. By allowing participants to build exploits and actually apply the knowledge they gained through the previous lectures, we hope they gained perspective on the tools they would build or use in the coming weeks to prevent similar intrusions from succeeding on their server.

In addition to the classes, a mailing list and wiki were set up to provide information and answer questions. After the release of the competition VM (both before and after lectures and lab), participants posed a number of questions about server configuration, tool use, and defensive and offensive strategies using these resources. We also held a *post mortem* session right after the competition to discuss the vulnerabilities that we introduced into the plug-ins provided and to give teams an opportunity to explain their strategies, including how they found and exploited vulnerabilities. We used this forum also to solicit feedback about the mechanics and implementation of the competition itself.

4 EXERCISE DESIGN

This section covers design decisions made while planning the CTF and its component challenges.

4.1 Target Selection

In order to design a CTF that carried an academic flavor, we realized that challenges based on compiled binaries would require an unacceptably large amount of prior knowledge and thus contradict our pedagogical goals laid out in Section 2. As such, we chose a CTF setting that would naturally allow the participants access to the source code of the system. An easy way to do this was to focus on web application security.

Having chosen the game genre, the next decision was selecting an open source or custom-written web application framework. We believed that the game would be more meaningful to the participants if we used realistic, commercial off-the-shelf (COTS) software during the CTF, since it would allow them to build reusable expertise for a popular software package that they are likely to encounter again elsewhere. With this in mind we set out to select a common web application framework that would enable our CTF to be educational, well-designed, and fun to play. After considering several candidates, the PHP-based Content Management System (CMS) *WordPress* [1] was selected as the CTF's base architecture.

4.2 Modular Game Design

One of the main requirements in selecting a web application framework was modularity. We needed a robust way to introduce new vulnerabilities that could be exploited by competition participants that could not be discovered by simple source code "diff". A plug-in architecture, especially one as flexible and extensively used as in WordPress, allowed us to create new functionality that featured carefully crafted but realistic vulnerabilities. At the same time, this architecture enabled us to provide the participants with the basic framework (i.e., LAMP server with WordPress) ahead of the competition without revealing any details of the plug-ins we were building. Finally, separating our challenges into different plug-ins enabled easy division of labor.

Plug-ins are used extensively, particularly in web applications. We felt that the dynamics of acquiring untrusted code, examining it for potential flaws, fixing the ones that can be easily found and providing some kind of sandboxing or code isolation as a fail-safe was a realistic strategy that system administrators might employ. Formulating a game around this dynamic enabled participants to practice several important practical computer security skills, including source code auditing, fuzzing or web application penetration testing to find vulnerabilities, patching code without removing functionality, and configuring appropriate sandboxing mechanisms.

4.3 Pre-release of Select Game Content

To enable students to create significant solutions we wanted to release as much of the CTF content ahead of the competition as possible, without releasing the challenges themselves. By selecting a modular framework, we were able to withhold all of the challenges explicitly written for the CTF while providing a virtual machine to teams a month ahead of time. Since the distributed VM was almost identical to the one deployed at the competition, participating teams were encouraged to build defensive tools and scripts to lock down their servers, while verifying the base server configuration was secure.

Furthermore, since WordPress has a large set of publicly available plug-ins, many of which have published security vulnerabilities, we were able to further augment the pre-released virtual machine with representative sample plug-ins. Three vulnerable WordPress plug-ins and corresponding exploit code were gathered from `www.exploit-db.com` and installed into the VM's WordPress instance. This provided a proxy for what the teams would see during the game, thus enabling teams to build and test defensive measures and sandboxing implementations before the competition began.

4.4 Simulating the Real World

In addition to supporting the pedagogical goals described above, we aimed to emulate the dynamic nature of real world operations. Business pressures require IT infrastructure to be nimble and provide new functionality on short notice. To simulate these requirements during the CTF, we released one batch of new plug-ins at the beginning of the competition, and another batch near the end of the first day. Because plug-ins provided independent functionality, teams could choose to run some of them, but not others, thus giving them time to review and harden new plug-ins; however, any delay in enabling plug-ins corresponded to sacrifices in the availability score.

4.5 Selecting Participants

Given the open nature of the MIT/LL CTF, we did not require any qualifications of our participants, aside from being enrolled in an undergraduate or graduate level program at a Boston-area university and willingness to spend the weekend competing in the CTF. We encouraged participants to form teams of at least three members, as we felt that competing with fewer people would put the team at a significant disadvantage. Our resulting participant pool was comprised of undergraduates from MIT, Northeastern University, Boston University, Olin College, University of Massachusetts (Boston), and Wellesley College, divided into (some multi-institutional) teams of three to five members.

5 Scoring

In order to evaluate the teams' performance during the CTF, we separately assessed each team's ability to defend their server (the *Defense* subscore) and to capture other team's flags (the *Offense* subscore). The defense portion of the score was itself derived from three measures: *confidentiality* (C), *integrity* (I), and *availability* (A) that were combined using weights W_C, W_I, and W_A.

$$\mathit{Score} = W_d \times \mathit{Defense} + (1 - W_d) \times \mathit{Offense}$$
$$\mathit{Defense} = W_C \times C + W_I \times I + W_A \times A$$

Varying values for W_d, W_A, W_I and W_C provided much-needed flexibility for simulating scenarios with different importance assigned to the corresponding properties. The rest of this section describes how each of the score components was measured.

5.1 Functionality-based Metrics

Each team's *availability* score was measured using a grading engine with a module written for each plug-in, with an additional module to evaluate WordPress's basic functionality. Scores from each grading module were combined into a weighted average, according to an assigned importance of each plug-in. For this competition, all functionality was weighted evenly; however, these weights could be easily adjusted to reflect the difficulty of securing a plug-in that provides some complicated functionality.

During the competition each team's website was evaluated on 5-10 minute intervals. A team's overall availability score was calculated as the mean of all availability scores to date.

5.2 Flag-based Metrics

Every 15 minutes, flags (128-character alphanumeric strings) were deposited into the file system and database on each team's server. If opposing teams captured these flags, they could submit them to the scoreboard system. Flags were assigned a point value corresponding to the perceived difficulty and level of access needed to acquire the flag. By providing flags of varying difficulty, we hoped that teams try to escalate privilege to gain higher levels of access than those afforded by the more basic exploits available in the game. Following the insertion of flags into each system, the scoring bot would wait a random period and check whether the flags were unaltered.

The *confidentiality*, *integrity* and *offense* score components were derived from the flag dropping and evaluation system described above. Integrity was calculated as the fraction of flags that were unaltered after the random sleep period; if the VM was inaccessible, a score of zero was reported, as the state of the flags could not be determined. The instantaneous integrity scores were averaged together to produce the integrity subscore.

Confidentiality and offense were closely linked portions of each team's score. Confidentiality was a running record of the percentage of flag points assigned to a team that had not yet been submitted by an opposing team for offensive points. Conversely, offense was calculated as the raw percentage of flags that a team did not own themselves but submitted to the scoreboard for points. Unlike availability and integrity scores, which were immediately computed by the grading system, the confidentiality and offense scores were based on flags submitted by different teams. Since a flag could be submitted to the scoreboard by a team any time after that flag entered the CTF ecosystem, it was impossible to instantaneously tell when a given team's confidentiality was compromised.

5.3 Situational Awareness

In an effort to provide situational awareness, the scoreboard presented two informational screens to each team. The *errors view* provided access to availability and flag rotation errors. Each availability error was tagged with the corresponding plug-in and a timestamp. Flag rotation errors indicated to the team when a flag could not be dropped onto their system, and whether it was a database or a file system flag. With both of these pieces of information, teams were adequately able to detect and fix broken functionality on their system.

The *grading view* provided each team with a breakdown of their scores in a more granular fashion. The last ten instantaneous integrity and availability scores were displayed to each team, enabling them to identify when their level of service had been adversely affected. Confidentiality was displayed as a number of flag-points not scored by other teams out of the total number of flag points assigned to the specific team. Conversely, offense was displayed as the total number of flag points scored out of the total number of flag points in the CTF not belonging to the specified team. We considered offering information about confidentiality and offense score in a similar fashion to integrity, but realized that data was not meaningful because of the practice of flag hoarding – i.e. collecting flags from an opponent's VM but not submitting them instantly to conceal the intrusion.

6 Data Analysis

In our CTF, we collected data from two different sources: the availability, integrity, confidentiality and offense scores aggregated during the competition itself, and participant surveys distributed after the competition. Through analysis of the scoreboard information we can gain insight into the way the scoring function affected and incentivized certain actions within the game, while survey responses provide a way to gauge effectiveness of the exercise in educating, challenging, and enticing participants into the field of computer security.

6.1 Scoring Data Analysis

The scoring function discussed in Section 5 included adjustable weights to enable us to shift the balance of the gameplay if we deemed the CTF to be too focused on some aspect of defense or offense. On day one, we set $W_d = \frac{1}{2}$ (thus giving even weight to defense and offense components), and set $W_{\{A,I,C\}} = \frac{1}{3}$. In this configuration, we discovered that the teams shut off the web server (or turned it on only during scoring runs), thus removing most obvious pathways an adversary would use to take over the system. This strategy was cost-effective, as availability was worth only $\frac{1}{6}$ of the total score, and the team's confidentiality and integrity scores improved with most paths to attack removed. Furthermore, this strategy enabled the entire team to focus on offense.

Since this was not the game equilibrium we were looking for, we adjusted the grading weights to shift the balance of the game back in favor of keeping the web servers running. On the second day, we announced that new scoring weights were $W_d = .80$ and $W_A = .80$, thus making availability worth now a hefty 64% of the total score. New weights, of course, did not apply retro-actively to scores obtained on the first day.

Given this change in scoring metric, we would have expected to see availability rise during the second half of the competition. We would have also expected resources from offense-related activities to be diverted to defense; thus, we expected to see offense scores decrease in the second half of the competition as well. However, the results showed a more complicated picture, as we now discuss.

6.1.1 Availability Scores

Figure 1 (left) shows the instantaneous availability for the top 5 teams (given the final standings).

The general trends in availability scores for the first day seem to imply that two strategies were in place. Some teams (DROPTABLES, Ohack, and Pwnies) seem to have enabled the web server and plug-ins at the beginning of competition and thus suffered from exploits throughout the day. Other teams (CookieMonster and GTFO) disabled the plug-ins at the beginning of the day, thus paying the cost in decreased availability; however, once they figured out some way to secure them, the web servers were turned back on.

On the second day, the picture is quite different. DROPTABLES and GTFO (1st and 2nd place finalists, respectively) battle to keep their availability up (presumably while under heavy attack). CookieMonster starts out with high availability, but drops rather precipitously as the team's web server is owned and eventually destroyed beyond recovery. Pwnies and Ohack struggle against attacks as well, but do manage to get the plug-ins functional towards the end of the competition. While we are not seeing a clear trend demonstrating that availability was higher during the second day of the competition, it is evident that the teams were trying to get their availability back up, even if some did not succeed.

6.1.2 Offense Scores

Figure 1 (right) shows the cumulative running average of the offense score for the two days of the competition (again, just for the top 5 teams). Because flag hoarding is allowed by our system, there is no "instantaneous" offense score – this score necessarily carries the memory of the

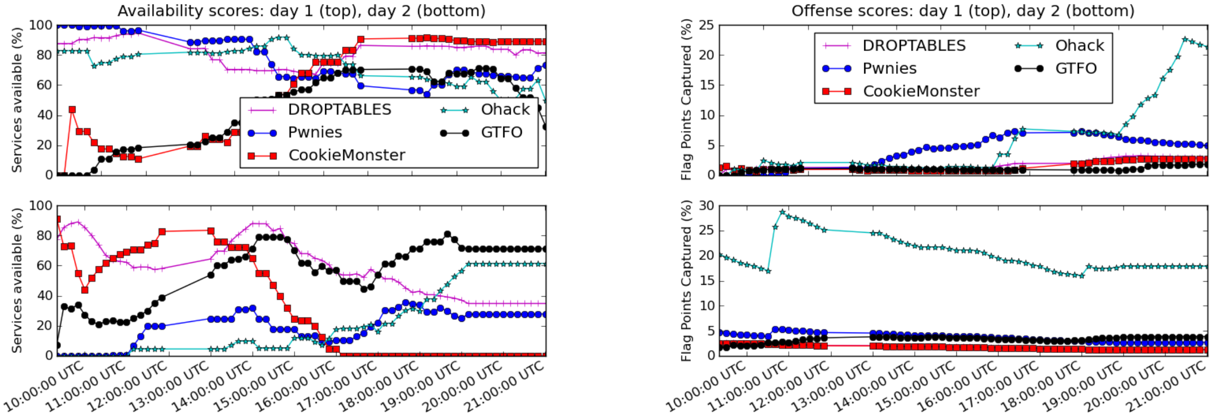

Figure 1: Instantaneous availability (left) and offense (right) scores for the top 5 ranked teams on day 1 (top) and day 2 (bottom). The plateaus between 13:00-14:00 and 17:00-18:00 represent lunch and dinner breaks, respectively. The competition ended at 21:00 on day 1 and 19:00 on day 2.

entire competition. Here, we expected to see a decrease in offense activity on day two, as teams' resources are redirected to defense to ensure highest availability.

On day one, Ohack and Pwnies dominate the offensive landscape (at the expense of their availability scores), while DROPTABLES, CookieMonster and GTFO spend less time on offense and thus provide higher availability (at least before the dinner break). On day two, Ohack again dominates all other teams in total number of flags submitted. The sharp jump for both Ohack and Pwnies coincides with our announcement that the scoring weights are changing – therefore, any flags that were being hoarded are submitted to the scoreboard as soon as possible. After that, offensive scores decline as the teams presumably spend more effort on defense, with the exception of GTFO, whose offense score increases throughout the day.

Overall, the data supports the notion that teams refocused their efforts from offense to defense due to the change in scoring metric. The results are somewhat muddled by the fact that the teams figured out several vulnerabilities by the time day two started, so maintaining a higher availability was more difficult.

6.2 Participant Survey Results

We released an Internet-based survey to MIT/LL CTF participants two days after the competition. This survey covered overall CTF impressions and education takeaways, pre-CTF class evaluation, evaluation of in-game strategies, and post-CTF reflections. The analysis below is based on 22 responses we received to the survey.

When asked to rate confidence in their computer security skills before and after the competition, we found an improvement of (on average) 1.4 points on a 10-point scale. Two respondents reported a decrease in perceived computer security skill, but their comments indicate that they may have been over-confident in their skill level before the competition began.

When polled about interest in a computer security career both before and after the event, respondents displayed an average of 1.1 point increase in interest (again, on a 10-point scale). This is likely due to the fact that many of those who responded stated that they had a 10 out of 10 interest in a computer security career even before the competition.

When asked to select the lectures that they found to be most helpful, the *Linux Server Lockdown* was the most popular (13 votes), closely followed by *Web Application Exploitation* (11 votes). Survey comments indicated that respondents found several techniques presented during our lectures useful. From the *Linux Server Lockdown* lecture, Apache's `mod_security` was mentioned as being helpful by several respondents. Additionally, respondents used `fail2ban` and `Tripwire` for detecting attacks, albeit some mentioned that `Tripwire` consumed too many system resources and had to be turned off. Surveys also indicated that knowledge of PHP's SQL escaping functions (described in *Web Application Exploitation* lecture) was useful in patching vulnerabilities in plug-ins. Respondents also reported using input "white-listing" technique taught in class to prevent command injection.

Of course, several knowledge areas that were not addressed in lecture proved very useful as well. Several respondents reported that being unfamiliar with PHP development posed a significant limitation – these teams were ill-prepared for auditing the plug-in source code to discover vulnerabilities via code inspection, which was useful both for patching and for exploit development. Aside from PHP development experience, some respondents indicated that remote system logging tools, specifically syslog-ng, proved useful for quickly detecting system attacks, and also in analyzing and re-purposing competitors' exploits. Finally, social engineering skills (an aspect not covered in lecture) helped one team obtain root access on several other teams' VMs. This "exploit" used an ingenious combination of a Gmail account resembling one used by the organizers, a signature block copied from previous announcements to the list of CTF participants, and a replica of the official CTF wiki complete with a not-so-authentic SSH public key to be used in the authorized_keys file of the root user on team VMs.

All survey respondents indicated that they spent at least an hour preparing for the competition. The two most common preparation time ranges were 1-2 hours (9 respondents) and 4-8 hours (8 respondents). When asked about time allocation between offense and defense during the competition itself, 50% of respondents reported that they spent more time on defense, 36% claim to have spent the majority of their time on offense, and the remainder indicated neither offense nor defense occupied the majority of their competition time. 86% of respondents indicated that they attempted to patch at least one of the vulnerable plug-ins, while those who developed exploits created an average of 1.5 exploits during the competition. Overall, 91% of those who responded said they would like to participate in an event like this again.

7 Related Work

Many other cyber security exercises, both with academic and recreational motives, have shaped the current CTF landscape. Perhaps the most notable is the annual DefCon CTF exercise held in Las Vegas, Nevada. This competition is open to participants world-wide, but has a preliminary challenge-based qualification round in which participants solve computer security puzzles on a Jeopardy-like board. The top eight teams from the qualifiers have historically played in a team versus team security competition during the DefCon conference, an experience described in detail in [5].

Notable academic CTFs include University of California, Santa Barbara's iCTF, the National Security Agency's Cyber Defense Exercise (CDX), and the Collegiate Cyber Defense Competition (CCDC). From 2003-2007, iCTF had a team versus team format similar to the MIT/LL competition. Since then, they have moved towards a storyline oriented model which provides each team with identical parallel versions of the game [4]. CDX places the emphasis on the defensive aspect of security, employing a *Red Cell* of hackers from various government organizations to simulate a live, defensive operation for the participants [13]. The organizations that compete in this game often hold semester-long courses to prepare for the exercise. Similar to CDX, CCDC also places the emphasis on the defensive aspect of security, employing an external third-party red team charged with attacking all participating teams.

Many other challenge or puzzle-oriented security exercises exist as well. One popular and notable example is NYU Polytechnic Institute's Cyber Security Awareness Week (CSAW) competition, which is loosely based on the DefCon CTF qualifier competition, but also includes a hardware security challenge and a game segment targeted at high school students. In addition to the many competitions held at the inter-collegiate and semi-professional levels, many other exercises have been held at universities [9, 12]. These events tend to vary in focus between defensive and offensive, and are often the capstone event of a semester-long course.

MIT/LL CTF is similar in some aspects to the aforementioned academic CTFs; however, our main goal was to make the CTF experience available to students of different academic backgrounds and varying practical expertise in computer security. By giving students an opportunity to experience what it is like to defend and attack computer systems first-hand as part of this competition, we were aiming to encourage interest in practical computer security and promote further study outside the competition. By providing a short series of lectures and labs before the event instead of requiring completion of a semester-long course we lowered the barrier to entry and enabled a wider group of students to participate in this event. The competition was not structured to protect teams with little computer security expertise (in fact, several survey respondents found the competition quite difficult and gained new respect for the skills required to succeed); however, we believe that it was accessible to the students who attended our lectures and prepared in the weeks preceding the competition.

Several self-paced security courses, including Google's Gruyere [7] and Stanford's Webseclab [3], also help students learn about web application security. However, we believe that a CTF based around defending an entire server in real time while running possibly buggy plug-ins allows students to learn about a broader range of computer security topics, as well as about operational considerations involved in maintaining a secure system.

8 Lessons Learned

Running the MIT/LL CTF was a new experience for all team members involved. Aside from learning quite a bit about the mechanics of web application exploitation while constructing our WordPress plug-ins (and wondering how the world of web applications can ever be secured), we also learned a few lessons about running a competition of this scale and complexity that are worth sharing.

8.1 Marketing and Scheduling

Running a CTF that includes participants from multiple universities required paying a lot of attention to scheduling and marketing. Originally, we planned on a week of day-time preparatory classes, followed by a week of competition during MIT's month-long break in January 2011. Despite extensive postering and e-mailing, we received very little interest, which was likely due to two causes: student were too busy to spend two weeks on this activity, and several other January classes and competitions (with significantly larger prizes) targeted students interested in web applications and security.

Feedback from MIT's SIPB and BU's BUILDS student groups helped us choose a better schedule by shifting classes to evenings and limiting the competition to one weekend. The new format was quite successful by limiting the time commitment required for a busy student. Advertising the CTF at multiple Boston-area universities also greatly increased participation – even though we required physical presence, students from Wellesley and Olin College braved the drive and came to compete anyway.

8.2 Infrastructure and Data Collection

We built our CTF infrastructure on VMWare's ESX solution, which afforded us quite a bit of scalability and robustness from the beginning. While we were careful in designing and testing our grading bots, the scoreboard and other "in-game" infrastructure servers, we still had a number of issues to fix during the competition itself. Some of the participants suggested that we hold scrimmages before the actual CTF to work out the kinks in the infrastructure, and that seems like a very good suggestion. Despite extensive testing using failure modes that we could easily predict (e.g., Apache is down, machine is firewalled, MySQL is not running), we ended up having to contend with unexpected scenarios. For example, our flag rotator bot relied on using `rm` command to remove previously-placed flags in the file system. In desperate attempts to stem damage from attacks, some teams decided to delete `/bin/rm` from their systems, resulting in completely unexpected grading bot failures.

A competition like a CTF is a great opportunity to collect data that might elucidate the ways an adversary actually attacks the web application, and what actions both attackers and defenders take throughout the competition. Our ESX server was setup to capture all network traffic that traversed the virtual switch; regretfully, something went wrong overnight, and the file system appeared to have corrupted our packet capture file. In retrospect, we would have liked better visibility into participants' servers, some of which might have been obtained using the ESX console or some custom software using VMWare tools API. For example, volume-based denial of service attacks were forbidden by our CTF rules, and we were asked several times by participants to mediate a situation where excessive network or CPU bandwidth was being utilized. Our methods used to verify validity of such claims were rather rudimentary – we could attempt to connect to the server in question, or traceroute the purported source of the denial-of-service attack; with some preparation, we may have had better tools at our disposal.

8.3 Operational Issues and Game Mechanics

When designing and implementing our grading bots, we were concerned about encouraging undesirable behaviors – e.g. if the grading bot is easy to identify, a participant might open the firewall only for the grading bot and exclude all other teams from accessing their server. Due to time restrictions in developing the grader bots, we used a different mechanism to interact with WordPress (via XMLRPC, used by desktop-based blogging software) instead of through a scripted browser. In retrospect, this was suboptimal, as certain activities performed by the grader were clearly distinguishable; furthermore, some aspects of the plug-ins ran JavaScript on the client and thus were not exercised by the grader. To make grading slightly less predictable, we did the following:

- Randomized starting time for a grading run (within a 10-minute interval),
- Randomly picked content drawn from 100 papers generated by MIT's Automated Paper Generator available at `http://pdos.csail.mit.edu/scigen/`,
- Randomly picked a string for the `'User-Agent'` header from a large set obtained from `http://www.useragentstring.com`.

In order to make the competition more accessible to weaker teams (and to obviate the necessity for teams to create their own whole-system backups), we offered each team three snapshots and three reverts per day of the competition. Because we couldn't give each team access to the ESX console, we performed these operations ourselves when requested by a team member. This "lifeline" enabled teams whose server was completely destroyed to continue participating in the game; however, the mechanics could be improved. We required a team member to approach us in person and bring a token given at the start of the CTF to identify their team in order to request a snapshot or a revert. Since all participants were in the same room, it was obvious that a certain team's VM will be momentarily reverted – in fact, requesting a restore became quickly known as the "walk of shame". Unfortunately, many teams created snapshots once their VMs were already infiltrated; thus, the attackers were able to compromise the machine again before countermeasures could be taken by the defenders. In addition, the initial snapshot provided to the teams had all plug-ins installed and Apache running, which again created a vulnerable situation and prevented the lifeline from being useful (especially during the second day of the competition).

8.4 Measuring the CTF's Educational Impact

To measure the effect of the CTF competition on participants' knowledge of computer security, we conducted an Internet survey after the event. The survey was rather *ad hoc*, as our primary motivation was to interest students in the field of computer security, to make the CTF experience available to those with little practical experience in the area, and to run an exciting competition. In these goals, the survey results indicate that we were successful. In future iterations of the MIT/LL CTF, we plan to measure the effect CTF had on participants' knowledge more directly by following methods similar to [10], by including pre-CTF and post-CTF quizzes to assess players' prior knowledge of computer security areas and experience gained by participating in the CTF. Since this is a completely voluntary game, we will likely need to incentivize participation in these quizzes to ensure statistically significant results; perhaps a raffle for a small prize would work well.

9 Conclusion

The MIT/LL CTF competition was a great learning experience both for the students involved and for the organizers. We believe that this exercise helped the students understand the intricacies of practical computer security, highlighted their strengths and weaknesses in computer security skills and generally increased their interest and desire to learn more about this area. We plan to continue fostering this community by encouraging creation of reading groups focused on practical computer security and by running similar CTF competitions in the upcoming years, incorporating the feedback from this year's participants.

Acknowledgments

We would like to thank the anonymous reviewers and our shepherd, Sean Peisert, for providing feedback that helped improve this paper. We would also like to thank Lincoln CTO, Bernadette Johnson, for her guidance and encouragement. Additionally, we thank Geoffrey Thomas of MIT SIPB for his presentation on securing Linux servers.

References

[1] WordPress: Blog tool and publishing platform. http://www.wordpress.org.

[2] S. Bratus. What hackers learn that the rest of us don't: Notes on hacker curriculum. *IEEE Security and Privacy*, 5(4):72–75, 2007.

[3] E. Bursztein, B. Gourdin, C. Fabry, J. Bau, G. Rydstedt, H. Bojinov, D. Boneh, and J. C. Mitchell. Webseclab security education workbench. In *Proc. of the 3rd Workshop on Cyber Security Experimentation and Test*, Washington, DC, August 2010.

[4] N. Childers, B. Boe, L. Cavallaro, L. Cavedon, M. Cova, M. Egele, and G. Vigna. Organizing large scale hacking competitions. In *Proc. of the 7th Conference on Detection of Intrusions and Malware & Vulnerability Assessment*, Bonn, Germany, July 2010.

[5] C. Cowan, S. Arnold, S. Beattie, and C. Wright. Defcon capture the flag: Defending vulnerable code from intense attack. In *Proc. of the DARPA Information Survivability Conference and Exposition*, Washington, DC, April 2003.

[6] R. L. Fanelli and T. J. O'Connor. Experiences with practice-focused undergraduate security education. In *Proc. of the 3rd Workshop on Cyber Security Experimentation and Test*, Washington, DC, August 2010.

[7] Google, Inc. Web application exploits and defenses. http://google-gruyere.appspot.com/.

[8] M. E. Locasto. Helping students 0wn their own code. *IEEE Security and Privacy*, 7(3):53–56, 2009.

[9] G. Louthan, W. Roberts, M. Butler, and J. Hale. The Blunderdome: An offensive exercise for building network, systems, and web security awareness. In *Proc. of the 3rd Workshop on Cyber Security Experimentation and Test*, Washington, DC, August 2010.

[10] M. Mink and R. Greifeneder. Evaluation of the offensive approach in information security education. *IFIP Advances in Information and Communication Technology*, 330:203–214, 2010.

[11] T. J. O'Connor, B. Sangster, and E. Dean. Using hacking to teach computer science fundamentals. In *American Society for Engineering Education, St. Lawrence Section*, 2010.

[12] M. O'Leary. A laboratory based capstone course in computer security for undergraduates. In *Proc. of the 37th SIGCSE Technical Symposium on Computer Science Education*, Houston, TX, March 2006.

[13] United States National Security Agency. Fact sheet: NSA/CSS cyber defense exercise - after exercise. http://www.nsa.gov/public_info/_files/press_releases/cdx_fact_sheet.pdf.