The Fun and Future of CTF *

Andy Davis, Tim Leek, Michael Zhivich, Kyle Gwinnup, and William Leonard
MIT Lincoln Laboratory
mitllctf-org@mit.edu

Abstract

Capture the Flag (CTF) is well-established as a computer security contest of skill in which teams compete in real time for prizes and bragging rights. At the time of this writing, CTFtime.org [4]—a tracking web site devoted to aggregating team standings across various CTF events—lists 76 such contests, and more spring up each year. But what is the point, exactly? In this paper we detail our experiences in a third year of designing, building and running a CTF for Boston-area undergraduate and graduate students. This will serve two purposes: first, others desiring to stage such an event can benefit from our experience, and second, the details of our CTF will provide a concrete context for a broader discussion and deeper questions on the value and future of this type of activity.

1 Introduction: What is CTF?

Several distinct kinds of Capture-the-Flag events have evolved over the years; however, most are a variation on one of three themes: attack-defend, attack-only, and defend-only. In this section, we give a brief summary and examples of each event type.

1.1 Attack-Defend CTFs

The DEF CON CTF is the original and, without question, the most prestigious capture-the-flag event [6]. This event requires participants to secure and operate a set of vulnerable services provided by the event organizers, as well as attack instances of the same services operated by other teams. Teams earn points for capturing secret information (*flags* [1]) from other teams' hosts by exploiting vulnerabilities in their services. Points are also awarded a team for ensuring that its services remain secure and function correctly. The latter is determined by automated querying of services (*polling* or *grading*) by the game infrastructure to verify service availability and functionality. For a description of the experience participating in this elite computer security event, we refer the interested reader to [2]. Attack-defend CTFs tend to be small in scale, with perhaps 10-20 teams, largely because attacking hundreds of teams is viewed as an orthogonal exercise in exploit automation.

1.2 Attack-only CTFs

In an attack-only competition, teams are only completing offensive tasks, and the organizers provide the infrastructure to host the target services (typically one per team). Teams once again earn points by stealing flags, which are retrieved by successfully exploiting a vulnerable service or solving a puzzle (generically, a *challenge*). A popular format for an attack-only CTF is a *Jeopardy-style* event, in which challenges are arranged by difficulty in different categories (e.g. "binary exploitation", "web", "cryptography", "trivia", "forensics", etc), much like the TV game show. Attack-only events can scale well to a large number of teams or participants (popular events have several hundred teams). DEF CON Quals and NYU Polytechnic Institute's Cyber Security Awareness Week (CSAW) competitions both use this format [6, 3].

1.3 Defend-only CTFs

Another common CTF offering involves only defense challenges. This game is typically targeted at high school students, college students and cadets at military service academies, with a single third-party team playing the offensive or "red team" role. Participants in such CTFs earn points by securing services on an already-deployed system (which may have been previously infiltrated), in

*This work is sponsored by the Laboratory for Telecommunications Science under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

[1] Typically, these are long, random, unguessable strings

addition to maintaining functionality and performing urgent "business inject" tasks that require system administration skills. Evaluation criteria vary between different events, and grading is frequently manual, not automated. As there are no flags and no capturing, it is perhaps odd to call this a CTF. A well-known event of this type is the Collegiate Cyber Defense Competition (CCDC) [12]. A high-school version called Cyber Patriot [1] has been gaining in popularity in recent years as well.

2 Why CTF?

Why do we organize CTFs, and why do people play in them? While we cannot speak for all organizers or players, we offer some answers in this section.

2.1 Why Run a CTF?

Aside from the obvious recruiting reasons, we run CTFs as a way to learn about how networks and hosts are defended and attacked. In principle, CTF provides an invaluable environment in which to study practical computer security. As the event organizers, we write the software that is to be defended and attacked, manually embedding vulnerabilities. Thus, we know the location of at least some bugs. Further, scoring demands that we also know precisely when services are up and functioning correctly, as well as when they have been successfully exploited, since flags stolen from services are submitted for credit. So we know when a team has been exploited, even if an attacker has decided to leave the service up in order to continue to rob it. Additionally, we can monitor network traffic and use flows to tell us which teams are interacting with which other teams at what times. We can even employ IDS with custom signatures tuned to detect exploits for the vulnerabilities we added. There are even more advanced possibilities: if defended servers are VMs subject to introspection of our construction, we ought to be able to trace and characterize temporal defensive activity. In short, we believe it should be possible to run a CTF and collect enough data over the course of the game to permit us to determine not merely which team won, but *how that team won*.

As an additional benefit, CTFs provide a sandbox in which prototype technologies (both defensive and offensive) can be tested and evaluated. Some of our CTF events have employed this kind of technology which has even been in-game. In these cases, CTF field-trials provided valuable data about security, usability and robustness.

2.2 Why Play in CTF?

We believe that people play in CTFs for one reason: *because it is fun*. We can think of no other explanation for the fact that 165 students from 10 local universities spent 48 continuous hours in a room at MIT actively engaged in computer defense and offense with us in October of 2013. Nor do we think this is a bad motivation. That said, we believe participants *ought* to be playing CTF because it offers them an opportunity to learn a great deal about practical computer security – just ask anyone who has lived through an attack upon his or her organization's computers if any important lessons were learned about computer security. CTF attempts to model that attack/defense experience. Time pressures bring into sharp focus theoretical lessons learned in class or absorbed from textbooks. Competitive forces exposes assumptions and flaws in techniques, tools, and systems constructed in a vacuum.

3 MIT/LL CTF

We have run three attack-defend CTFs on MIT campus, in 2011 [16], 2012, and 2013. To provide context for discussion, we will describe our most recent event in some detail. It should be noted that this infrastructure is now fairly stable, and has been used, unmodified, to run three successful "mini-CTFs" in 2014 already. Our goal is to open-source this codebase within the year.

In the 2013 MIT/LL CTF, each team played the role of a mobile app developer who was in charge of supporting several different apps written for the Android platform and corresponding back-end services running on Linux VMs. In order to distribute the app, each team was required to upload it to a public MIT/LL CTF app store, from which grader programs running emulated Android devices could download it. Teams were graded end-to-end on availability of app functionality and thus also back-end service. As part of that grading, new flags were deposited, and integrity of previously-deposited flags was verified. Confidentiality was measured indirectly – the flags that were stolen by other teams from apps or services were submitted to a Black Market for credit. The team with the highest score at the end of the game was declared the winner; prizes were awarded to the top three placing teams.

3.1 Game Architecture

Our game architecture is depicted in Figure 1. Players were in charge of administering, operating, and defending their Team Servers. Attacks were to be launched from player laptops, which were not to be, themselves, attacked. All other infrastructure components (Score-

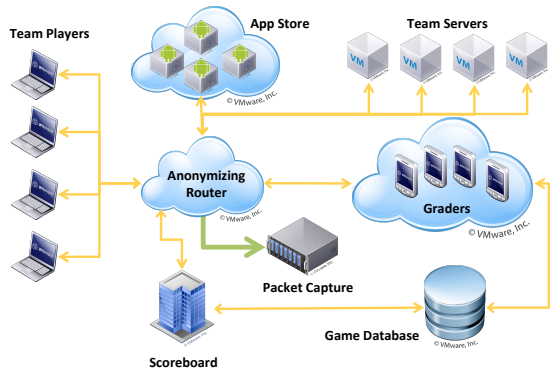


Figure 1: 2013 MIT/LL CTF Game Architecture.

board, Game Database, Graders, Anonymizing Router and App Store) were operated by the organizers. The role of each of these components is described in more detail below.

3.1.1 Team Servers

At the beginning of the game, each team was given access to a virtual machine with an archive containing the challenges. Teams had administrator access to the VM and were encouraged to set up whatever defenses they deemed necessary. Players were also given limited ability to snapshot and restore their VMs. We have found that providing this life-line makes the game much more accessible to more novice players.

3.1.2 Network Anonymization

Players connected to the game network through an *anonymizing router*, which hides their source IP address. The router doubles as an IDS and packet capture device for the CTF organizers, enabling us to analyze network traffic for the entire game. Experienced teams heavily monitor network traffic on their servers both to collect attacks for replay but also to characterize grader traffic in order to filter it effectively. The anonymizing router is necessary to prevent a players from identifying IPs of graders, whitelisting them, and simply blocking everyone else. Despite this, some teams in the 2013 event reported afterward that they were able to distinguish grader traffic by TCP source port number. This is a great example of the arms race between CTF teams and organizers; next time, we will be randomizing source ports.

3.1.3 Graders

Grader programs tested each team's challenge services and apps to make sure they were functioning and retaining information correctly. Every round, with a random delay, a grader performed the following actions for every team and every challenge:

Deposit: A new flag was generated and deposited into the challenge.

Availability: The challenge was checked to make sure that all important functionality was present.

Deposit Stolen: Every flag that a team stole and submitted for credit in the previous round was deposited into the stealing team's service.

Integrity: A previously deposited flag was selected and the grader verified it could still be retrieved.

There are at least two interesting consequences of this grader design. First, a CTF challenge service must act as data repository, collecting more flags the longer it is available. Second, scoring points for stealing flags from another team for a particular service requires the stealing team to operate the same service in order for *deposit stolen* to succeed. We additionally arranged for flags to expire after a certain number of rounds, after which they do not earn points on submission. This guards against flag-hoarding which makes the game less dynamic.

Since graders in our scenario represented mobile phones running Android, the graders were set up periodically to download team apps, install them on an emulated phone and interact with them to verify correct functionality. Apps for several teams co-existed on the same device, thus creating a playground for automated exploitation; several vulnerabilities were included in the Android app challenges that would enable an observant attacker to intercept communication and steal flags on device, without the need to attack the corresponding back-end services and risk exposure.

Android emulation was enabled by PANDA, our Platform for Architecture-Neutral Dynamic Analysis [11]. PANDA includes full support for Android via a careful port of critical aspects of the Google Android emulator, and also provides snapshots, record/replay, and analysis plugins. Android app graders were implemented with the UIAutomator testing framework [8] for which we wrote small Java programs that would exercise apps in complicated ways, deposit flags, and perform integrity checks.

3.1.4 App Store

The App Store for our CTF made available apps to be installed on emulated phones for grading. Since the app store was accessible to both graders and other teams, modifications (whether patches, security features or automated attacks) that were made to the Android apps were thus also available to other teams for analysis.

This was a significant departure from traditional CTF gameplay, where fortifications and attacks are not readily available to opponents.

Another novel aspect of the game was our use of a research technology to secure the App Store.

We used the MIT/LL CTF as an opportunity to conduct red-teaming of the Advanced Adaptive Application Environment (A3) [13] technology built by BBN as part of the DARPA CRASH project [5] by letting the students attack the App Store. The App Store presents a high value target for teams since the App Store holds all of the team's apps.

The App Store consisted of a simple PHP application that required a password to upload a new Android package for a particular app/team pair using a database and a file system to store the actual package data. The app store code contained several intentional vulnerabilities, including directory traversal, SQL injection and OS command injection. The A3 team was given the daunting challenge of securing the app store without modifying the vulnerable code. To do this, they employed input filtering, virtual machine introspection, and application proxies to enforce a security policy.

The A3 team had mixed success in defending the app store; early in the game, CTF players managed to bypass A3's input filtering policy due to a software bug in the policy implementation mechanism. A3 was down overnight until software engineers were able to identify and repair the flaw the next morning. During this time we switched to a manually hardened version of the app store, so that the game could proceed. Once these issues were resolved, the A3 app store continued to operate properly for the remainder of the game. Overall, this was a very useful experience for us as organizers and for the A3 team as participants – it showcased CTF as a venue for testing new technologies and it cemented our view that this is only a viable option if technology developers are on hand to maintain and fix their prototype. The A3 team walked away with fewer bugs in their code, validation of their defensive policies, and a corpus of attacks that they could use in further development.

3.1.5 Scoreboard

The Scoreboard was the player's primary interface to the game. Before the event, players registered with the scoreboard and formed teams. The registration process enabled us to ask survey questions of players, including demographic information, prior experience, and, most importantly, T-shirt size. During the game, the team captain could access credentials, snapshot / restore the Team Server VM, and submit stolen flags. Each round, graders would send messages to a team to provide diagnostic information about scoring; this information is typically limited to pass/fail status of a particular service. As the name implies, the scoreboard also displayed the current score at the end of each round. When players registered for the game with the scoreboard, we recorded IP addresses of incoming connections, which enabled us to cross-reference this information with our logs during the game to determine who was launching which attacks and breaking rules against flooding attacks.

3.1.6 Game Database

The Game Database is the brain of the entire operation; it was in charge of keeping track of participating teams, what flags were created, which team they were originally deposited to, and who submitted them for points. Our design uses an append-only transaction log of all operations within the game, enabling us to not only compute score on the fly but also perform analyses on all actions taken. Since the database is omniscient, we were able to implement sanity checks that verified validity of the transactions (that is, they matched our expectation about order, number, etc). These checks have been instrumental in finding and rectifying bugs within this distributed system that would have been difficult to find otherwise. As a preventative security measure, since at least some access to the Game Database was granted to the Scoreboard web application, we employed heavy use of PostgreSQL stored procedures and principals/privileges provided by the database.

3.2 Scoring Philosophy

In any game, understanding how scoring works is crucial to winning. Conversely, when designing and running a game like CTF, a careful choice of scoring affects everything from fun to player incentives. Our primary goal in scoring was for the winning team to demonstrate a balance of defensive and offensive skills, in our experience, this maximizes both fun and learning. If we weight offense too highly, teams either turn off or do not bother patching their services. Both strategies make the game less interesting: in the former case, there's nothing to attack, so there is no game; the latter case, paradoxically, makes offense easier, since no one is defending anything. On the other hand, if scoring overly focuses effort on defense, then teams spend all of their time patching, jailing, chrooting, and proxying their services to defend against attacks that never come. The first several hours of this kind of game are extremely boring as no one wants to turn on their services until they are sufficiently secure.

3.2.1 Our Scoring Algorithm

There is, of course, no "best" solution – there are many CTF events every year and each seems to have its own

scoring ideas. Even annual events such as DEF CON CTF change scoring system from year to year [7, 9]. However, after years of debate and tweaking, our scoring algorithm and function has stabilized, and has the following desirable properties:

- Scores are monotonically increasing.
- Teams score only if their services are up.
- Teams that are preforming better offensively are more lucrative targets.

We believe that monotonically increasing scores make a game easier to reason about and more rewarding. Many games work this way, from Baseball to Scrabble. When scores only ever increase, high scoring dynamic games are rightly distinguished from low scoring slow-moving games. This kind of game is also less of a bummer; no one can take your home run away. Requiring that team services be up in order to score points both offensively or defensively provides a very strong incentive for every team to risk running services as soon and as much as possible. This absolutely makes the game more interesting as there is more to attack at all times. Further, successful offensive teams are also ones that actually run services (since stolen flags must be deposited in the stealing teams' services). This makes them obvious and juicy targets since they are sitting on large banks of deposited flags ready to be stolen.

Our scoring algorithm works as follows. The game is broken into rounds. A team's score is the summation over rounds. A team's round score is the summation over challenges. A challenge score is the product of the challenge availability and integrity, finally multiplied by the number of flags deposited into the challenge thus far (both legitimate and stolen). The score for team t after R rounds for C challenges is computed as in Equation 1, where $A(t,c,r)$ and $I(t,c,r)$ are the availability and integrity scores for challenge c in round r. $F(t,c,r)$ is the total number of valid flags held by challenge c in round r.

$$score(t) = \sum_{r=1}^{R} \sum_{c=1}^{C} A(t,c,r)I(t,c,r)F(t,c,r) \qquad (1)$$

Availability and integrity checks result in either a zero or a one. Therefore, if a service is down or its flags have been removed or modified (stealing a flag does not remove it from a team's service – instead, its value is effectively diluted), the team is punished heavily, receiving zero points for that round. This may seem harsh but we have found it to be necessary to counter the naturally tendency for many teams is to ignore defense and focus on the "more fun" offensive aspect of the game.

We will not try to argue that our scoring is simpler or more realistic than others, because it isn't. It requires a double-summation to write out precisely, after all. We do, however, believe we have evidence that it works. That is, it gives top ranking to a team skilled at both offense and defense. Figure 2 presents graphs of availability scores (on top) and cumulative flags stolen (on bottom) for three teams that played in our 2013 CTF. These are the only teams that displayed any significant offensive as measured by flags stolen. The winning team is indicated by lightest grey line in both plots. It was, by a good margin, the best team offensively. It was also reasonably close to the best team defensively. The team that won was, thus, precisely the team we felt deserved to win, a team that demonstrated sustained offensive and defensive skills over the course of the competition.

4 CTF Questions

A number of questions naturally occur to those organizing a CTF. In this section we will pose several of these questions in the context of attack-defend CTFs (about which we are most knowledgeable) and, when possible, attempt answers.

Question: Can CTF teach computer security? This is unknown. We have supplemented CTFs in the past with lectures and even labs, but they appear to provide little benefit. Those who attend these more traditional educational venues do not appear to be better prepared for active game play. This can be frustrating, and we often are asked: How do I improve at CTF? The answer to this question appears to be that to get better at CTF you should play CTF a lot, in much the same way that you play baseball a lot to become a better ball player. However, it is also clear that the more detailed domain knowledge you bring to game day the better. If you already understand how heap memory allocation works you have a head start if the vulnerability involved in a challenge is a use-after-free one. And if you already know x86 assembly you will be better equipped to reverse-engineer a Windows binary.

So where is the education in CTF? We believe CTF works as a kind of group self-guided project-based instruction. Participants teach themselves relevant computer security concepts and skills on the fly and under pressure in order to perform necessary offensive and defensive tasks. If you walk around the room during a competition, this means you will observe participants reading web pages, in discussion with team members, and building tools to solve immediate problems. That is, you will see them working to solve problems for which they do not already have ready answers. CTF provides players with a safe place in which to engage in what would oth-

erwise be risky business. They can attack services of others as a way of understanding, very concretely, how attacks work, without worrying about being arrested and prosecuted. And they can defend a toy service for a day and not worry too much because it isn't their business and thus livelihood on the line.

Question: Should CTF be realistic? If the goal of CTF is to teach real world computer security skills, then one would assume that the more realistic the game the higher the educational value. However, there are at least two ways in which CTFs of the attack-defend variety depart considerably from reality. The first is the compressed time frame, which, in our opinion, is necessary but not detrimental. The second is a requirement that teams engage in both offense and defense. Both we and DEF CON require this, and while it may be unrealistic, we believe reality is in error.

CTFs tend to take place over a one or two day period during which many participants barely sleep. This is likely unavoidable for scheduling reasons alone. Participants can clear a weekend for an event like this but are unlikely to be able to set aside a week or month. The resulting event is probably more intense and exciting than most equivalent day-jobs, but we don't think that is a bad thing. The time frame restriction may make it difficult to employ slow stealthy attacks or social engineering techniques (though the latter was successfully employed in our 2011 CTF [16]), and may encourage manual defensive solutions that do not scale well to real-world problems. We believe that despite these artifacts, attack-defend CTFs represent a reasonably realistic laboratory in which to practice offensive and defensive strategies.

It is our belief that requiring both attack and defense of the same team greatly enhances educational value as well as making CTF more fun. If defenders do not understand what they are defending against then it will be hard for them to succeed. Conversely, if attackers do not understand how defenders are protecting their systems then they will not understand why their attacks fail. We are actually hopeful that this idea may catch on, that reality may decide to emulate CTF. What if it turns out that the best defenders and attackers prove to be those who have significant experience engaging in both activities roughly equally? This may be accomplished by regularly rotating security practitioners between defensive and penetration testing roles throughout their careers.

Ultimately, we believe that CTF events should be realistic, but not be slaves to realism. They should not compromise educational value or fun at the expense of realism. If we succeed, then players will have plenty of real life challenges awaiting them after graduation.

Question: Does CTF work as a testing ground for new ideas? CTF events can provide a safe place to test new technologies and strategies. It is certainly a safer place to test than in the real world. We think the inclusion of A3 in our 2013 event was a very positive experience and much was learned.

However, relying on any experimental technology for a major event is risky. Even though we tested with the A3 team, we still had to have a backup solution (a manually secured version of the app store) in case the primary needed to be taken down. It seems it is best to deploy experimental technologies in secondary parts of the game that are not controlled by the players (e.g. we deployed A3 on our app store). This mitigates against two major issues. First, it allows you to easily swap out the technology if it fails, thus allowing the game to continue. Second, if it works too well and prevents all attacks then the game can continue because players are still able to just attack one another.

One very effective defensive technique that has materialized seemingly spontaneously in various forms in our CTFs is the use of proxies and input filtering for defense. Several open source and commercial solutions exist that employ this technique, and some academic work has investigated it in earnest [10]. This is extremely effective, at least in the context of our CTF. It is possible that this is due to the relative simplicity of our challenges, the low volume of traffic (allowing a human to analyze all of it), and the compressed time frame. We will be investigating ways to challenge and possibly quantify the effectiveness of this strategy in the future.

5 The Future of CTF

It is apparent to us that CTFs are growing in popularity. This seems to be simply because they are fun and not due to any real or perceived pedagogical value. There has been some debate, lately, as to the value of defend-only CTFs [15, 14]. Our position is that these events ought to incorporate an offensive component or they will be replaced with more fun and (we believe) more educationally valuable attack-defend style events. Certainly, this type of event is not on the rise. Rather, out of the multitude of CTFs that have sprung up of late, the vast majority are online and attack-only. We have several hypotheses as to why this is happening.

Scale: Popular attack-only CTFs get as many as a 1,000 teams signing up to play (though it is unclear how many actually participate). Regardless, it would require a tremendous amount of resources to host an attack-defend or defend-only CTF of anything like this scale. This is because, for every team, we need one or more VMs for them to lock down; that is, the requirement of a defense aspect makes the game

scale with the number of teams. Attack-only CTFs, on the other hand, can make do with a few servers per challenge. This ability to scale is why this type of CTF is always used as a qualifier for expensive attack-defend events like DEF CON, and being a qualifier only draws bigger crowds.

Fun: Students tend to enjoy attacking and view defense as somewhat of an annoyance [2]. This reflects real life where the star athletes on a team tend to be the ones who score points. And consider this: Anyone who wants to practice defense has only to stand up a server on the internet and wait for the attacks; conversely, practicing attacks on the real internet will get you arrested.

Player stress: Playing in an attack-defend CTF is stressful for the simple reason that your services are under continuous attack. We believe this stress has educational value. On the other hand, attack-only CTFs are unquestionably more relaxed affairs – you can walk away for a while without consequence; no one is going to bring your server down if you take an hour off for dinner.

Organizational resources: Building and organizing an attack-defend CTF is stressful, which means few are willing to do it. This kind of game is difficult to build and run successfully, the primary issue being the Rube-Goldberg machine that is responsible for automated grading. Further, putting on an event with about 150 people for 48 hours straight is difficult logistically, requiring space, food, 24-hour staffing, etc.

All of these factors contribute to the trend, but we have no reason to believe it is irreversible. We are working on ways to both scale attack-defend events better but also make them less stressful for everyone involved. Competition is healthy and we aim to win people over to our kind of CTF.

Where else do we want to see CTF go in the future? It is already a powerful education, outreach, and recruiting tool, in our opinion. But it can be much more. We believe CTF can enable scientific inquiry in new and exciting ways. As we indicated in Section 2, CTF should be an ideal environment for the study of practical computer security. We already know enough about what happens during the game to be able to pronounce, instantaneously, who is winning. As researchers, we would like to understand better what teams do during CTF events how effective their actions are. With introspection, we may even be able to answer interesting questions like *how a team won*. It may be possible to discover new advanced techniques for attack and defense by providing college students a safe place to play.

CTFs are certainly in vogue at the moment. It is unclear if they will remain as popular as they are today. They are an engaging vehicle and it is our job to figure out how best to harness that interest.

[2] If we are honest, so do several of the authors of this paper

Acknowledgements

The authors would like to thank LTS for being a patron of the MIT/LL CTF. We are indebted to Vern Rivet for his awesome sysadmin-fu, James Hardy for writing Android challenges, and Harry Phan and Diane Staheli for putting together groovy network and scoring visualizations. We would also like to thank Dinara Doyle and all the Lincoln volunteers who kept us and the CTF participants well-fed and caffeinated throughout this 48-hour hackathon.

References

[1] AIR FORCE ASSOCIATION'S CYBERPATRIOT. CyberPatriot: History. http://www.uscyberpatriot.org/about/history.

[2] COWAN, C., ARNOLD, S., BEATTIE, S., AND WRIGHT, C. Defcon capture the flag: Defending vulnerable code from intense attack. In *Proc. of the DARPA Information Survivability Conference and Exposition* (Washington, DC, April 2003).

[3] CSAW. CSAW CTF: About. https://ctf.isis.poly.edu/about/.

[4] CTFTIME.ORG. CTFtime.org/CTFs. https://ctftime.org/ctfs.

[5] DARPA. Clean-slate Design of Resilient, Adaptive, Secure Hosts (CRASH). http://www.darpa.mil/Our_Work/I2O/Programs/Clean-slate_design_of_Resilient_Adaptive_Secure_Hosts_(CRASH).aspx.

[6] DEF CON. DEF CON CTF Archive. http://www.defcon.org/html/links/dc-ctf.html.

[7] DIUTINUS DEFENSE TECHNOLOGIES CORP. About CTF. http://ddtek.biz/about-ctf.html.

[8] GOOGLE. UI Testing — Android Developers. http://developer.android.com/tools/testing/testing_ui.html.

[9] LEGITIMATE BUSINESS SYNDICATE. Finals 2013 Rules. https://blog.legitbs.net/2013/08/finals-2013-rules.html.

[10] LONG, F., GANESH, V., CARBIN, M., SIDIROGLOU, S., AND RINARD, M. Automatic input rectification. In *Proceedings of the 34th International Conference on Software Engineering* (Zurich, Switzerland, June 2012).

[11] MOYIX. Platform for Architecture-Neutral Dynamic Analysis. https://github.com/moyix/panda.

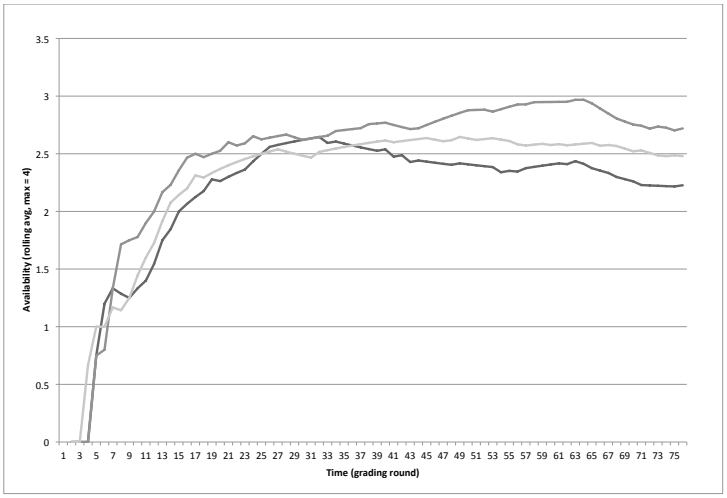
[12] NATIONAL COLLEGIATE CYBER DEFENSE COMPETITION. About CCDC. http://www.nationalccdc.org/index.php/competition/about-ccdc.

[13] PAL, P., SCHANTZ, R., PAULOS, A., BENYO, B., JOHNSON, D., HIBLER, M., AND EIDE, E. A3: An environment for self-adaptive diagnosis and immunization of novel attacks. In *Proceedings of Adaptive Host and Network Security Workshop* (Lyon, France, September 2012).

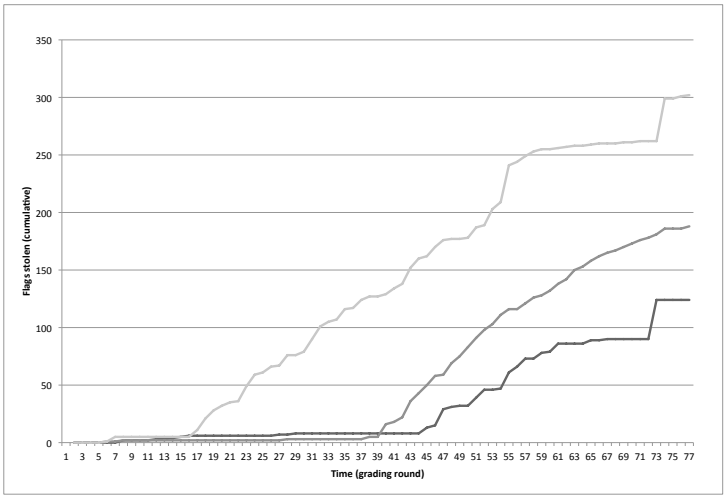
[14] PLAID PARLIAMENT OF PWNING. Why CTF. http://ppp.cylab.cmu.edu/wordpress/?p=1182.

[15] WEEKS, M. CCDC and CTFs – Addressing the Criticisms. http://www.scriptjunkie.us/2014/03/ccdc-and-ctfs-addressing-the-criticisms/.

[16] WERTHER, J., ZHIVICH, M., LEEK, T., AND ZELDOVICH, N. Experiences in Cyber Security Education: The MIT Lincoln Laboratory Capture-the-Flag Exercise. In *Proc. of the 4th Workshop on Cyber Security Experimentation and Test* (San Francisco, CA, August 2011).



(a) Running average of availability scores for top 3 teams. There were four challenges, so the average $\in [0,4]$



(b) Cumulative number of flags stolen by the top 3 teams.

Figure 2: 2013 MIT/LL CTF scores