# Learning Obstacles in the Capture The Flag Model

Kevin Chung, *NYU Polytechnic School of Engineering*
Julian Cohen, *NYU Polytechnic School of Engineering*

## Abstract

Capture The Flag (CTF) competitions have been used in the computer security community for education and evaluation objectives for over a decade. These competitions are often regarded as excellent approaches to learn deeply technical concepts in a fun, non-traditional learning environment, but there are many difficulties associated with developing and competing in a CTF event that are rarely discussed that counteract these benefits. CTF competitions often have issues related to participation, quality assurance, and confusing challenges. These problems affect the overall quality of a CTF competition and describe how effective they are at catalyzing learning and assessing skill. In this paper, we present insights and lessons learned from organizing CSAW CTF, one of the largest and most successful CTFs.

## 1. Introduction to Capture The Flag

Capture The Flag (CTF) competitions have been used in the security community as tests of offensive security skill since at least 1996. The first DEFCON CTF was in 1999 and many other CTF competitions have spawned since then, including CSAW CTF, PlaidCTF, PicoCTF, UCSB iCTF, Ghost in the Shellcode, Codegate, Boston Key Party, 30c3, and RuCTF amongst others. There are now more than 70 annual CTF competitions tracked by CTF ranking site ctftime.org. Each of these competitions seeks to realize different goals but at its core, CTF has always been about education and skill evaluation [3][4][5][6]. CTF competitions aim to bring security community members from all skill levels together to share information and to see who among them can solve the most challenges. The community has a huge appreciation for CTFs as a learning platform, but very few members have directly addressed the pitfalls and obstacles associated with developing and participating in CTF competitions.

For reference, CTFs are commonly structured with a group of judges/challenge developers led by a competition organizer. Judges create challenges that are then given a point value and placed online to be played by the public. Challenges are made available on a game board that lists the challenges, their point values, descriptions, and solve counts. Alongside the challenges, there is typically a scoreboard that lists teams' points and comparative ranking. Also typically included in a game is an IRC channel in which competitors discuss the competition and interact with the.

At its core, the CTF is a type of "open-book test" revolving around computer security. This lends itself to testing competitors on very obscure types of knowledge. The technical knowledge that is communicated during a CTF is often only mentioned in passing in documentation or a classroom. For example, the Python documentation briefly, and without much discussion, mentions that the pickle module is "not intended to be secure against erroneous or maliciously constructed data." This ambiguity was leveraged in the 2012 Zombie Reminder (Hack.lu CTF) and Minesweeper (29c3 CTF) to create challenges that involved un-pickling untrusted data. The Zombie Reminder challenge from Hack.lu asked teams to break a simple hash to load arbitrary pickled data while the Minesweeper challenge asked competitors to break an XOR cipher to load arbitrary pickled data (See Figure 1) [7]. These challenges serve as examples of how CTF competitions can introduce and evaluate security concepts that are not often covered in traditional academic settings, or in documentation.

```
def load(self, data):
   self.__dict__ = pickle.loads(data)

def save(self):
   return pickle.dumps(self.__dict__, 1)
```

Vulnerable source code from the 29c3 Minesweeper challenge

```
msg = "cos\nsystem\n(S'/bin/sh'\ntR.'\ntR."
msg = pickle.dumps(Exploit())
h = hashlib.sha1()
h.update(msg)
msg = "4n71cH3aT" + h.digest() + msg
tmp = ""
```

**Figure 1: Snippet of exploit for the 29c3 Minesweeper challenge**.

Due to the unstructured nature of CTFs, students typically learn individually and must directly apply the knowledge they've gained to succeed in a CTF competition. A common and important aspect of CTF is that it is often not possible to earn "partial credit" for a challenge: the challenge is either solved or not, resulting in either all or none of the points. By presenting challenges in this way, competitors are forced to continually modify their approach until they succeed, providing multiple opportunities to learn about a specific technol-

ogy. This continuous development and learning not only allows challenge developers to use CTF as a platform to teach competitors about a particular area of security, but also allows organizers to identify competitors with extremely driven personalities. Even if a driven individual is not technically competent enough to score well in a CTF, we believe that their tenacity would resolve that problem given enough effort and study. It is our belief that students who are willing to immerse themselves in the studying and culture that's associated with CTF will eventually succeed and build worthwhile habits for furthering their own knowledge.

### 1.1. Learning in Capture The Flag

CTF is not the only alternative means of competitive security education currently in use. Other examples of "gamified" learning include the Collegiate Cyber Defense Competition (CCDC) and CodeAcademy.

CCDC is one such example where a student team (blue team) must exclusively defend their network against a team of professional attackers (red team). CCDC's main focus is communicating the administrative aspects of defending a fictional corporation from attack, leaving aside many of the technical aspects of computer security. Additionally, CCDC fails to properly model the situations that it aims to [11]. The inherent imbalance of power between the red and blue teams result in a disparity between the benefits enjoyed by each. For example, the notoriety associated with being a part of the red team is far more desirable than being part of the blue team.

In comparison, the CTF model focuses entirely on self-education by providing teams the technical problems that they must solve. Instead of worrying about administrative tasks, the competitors focus entirely on the problem. Similarly important, notoriety is gained based on your success in the CTF, not by being chosen to be on a team.

CTFs also provide a truer game-like experience, and in particular, provide incremental feedback. The process of solving a challenge, submitting the answer, and knowing whether the answer is correct or not provides competitors feedback on where they stand in the game and where they stand on their current problem. We believe these incremental rewards help motivate competitors and keep them engaged throughout the competition's duration. This is in contrast to CCDC, which only provides feedback at the end of the competition.

In addition to feedback, CTFs often have a difficulty estimation associate with a challenge giving some indication of how difficult a challenge will be. This is similar to how CodeAcademy rates its problem sets. This allows teams to strategize; self-selecting challenges that they believe they can solve immediately or with some effort. CCDC doesn't share this property as problems are handled as they come instead of giving the players the choice of which problems to solve. We believe, that the CCDC approach is more inline with the "real world," but for learning purposes, allowing competitors to choose their learning is preferred.

## 2. Problems in the Capture The Flag Model

Despite the claimed success of CTFs as both an educational and skill evaluation platform, there are still problems that hamper the success of the model. One of the most important problems seen by us, as organizers, is the difficultly for a newcomer to start playing and become immediately immersed in the competition. In our experience, potential competitors are introduced to CTFs by meeting a security professional (or fellow student who tells them about an upcoming event, and that they should participate in to get more exposure to the security industry. However, it becomes difficult to continue playing CTFs if the competitors are unable to solve problems or independently further themselves in some way. Competitors that cannot solve any challenges will quickly get discouraged and stop playing. However, we believe CTF competitions have the potential to provide a scaffolding to help build player confidence, and a reward system to encourage player engagement. Here we discuss some of these deficiencies, as well as how CSAW has attempted to overcome them.

### 2.1. Difficulties in Playing

The qualities that make games easily playable (e.g. those that are self-describing, have low skill prerequisites, and are language neutral) are often not found in many CTF, meaning CTF is not easy to immediately pick up and play. Other games, like Checkers and GoFish, have very low initial barriers to entry and, as a result, are popular worldwide. A Checkers player does not need to research the game and understand fine technical details in order to win a game. The same does not hold true for CTF where deep knowledge of technology assists greatly with participation and success. In this way, CTF is more like Chess, where even if the basic rules are straightforward, they may take many years of experience to master.

Unfortunately, many CTFs are not designed with the beginner in mind, relying on heavy technical requirements that lead to beginners quickly becoming stuck and, ultimately, giving up. To a newcomer, the first step is often the hardest. The motions that become second nature to a CTF veteran are not always easily understood or accessible. Specifically, CTF challenges often

revolve around complex, highly technical skills, such as byte code de-compilation (DotNet CSAW CTF 2013), machine code disassemblers (Impossible CSAW CTF 2013), and file format reversing (ArmorAll GitS 2014 Teaser); all technical topics that are expressly designed to get even skilled competitors out of their comfort zone.

### 2.2. Challenge Design's Relation to the Success of the Challenge

The success of a CTF event, in terms of learning and evaluation, is directly related to the design of the competition's challenges. It's unlikely that an overly simplistic challenge will provide much educational value, but it will likely be solved quickly. Difficult challenges have the opportunity to teach more to a competitor, but risk alienating many competitors. Because their more skilled teammates are deeply concentrated on solving a challenge, beginners may be relegated to independently searching for ideas on how to proceed with a challenge. As such, good challenges provide hints towards their own solution process. One metric for measuring challenge difficultly can be seen by the number of times a challenge has been solved. An easy challenge will have a high number of solves but a difficult one will not.

A target difficulty must be set for the CTF that allows the organizer to properly assign point values and filter or modify challenges that don't meet that target. Different CTF competitions target different audiences and as such have difficulties tailored towards that audience. CSAW CTF is oriented towards undergraduates just beginning to learn computer security topics, and thus features introductory challenges. DEFCON CTF is meant to be played by offensive security veterans and is comprised of much more difficult challenges. The difficulty of every competition's challenges are generally oriented specifically for their respective audiences and thus, provide challenge designers with a consistent target for difficulty designing challenges for their target difficulty maintains this consistency.

Very often organizers attempt to make a challenge harder by making it more convoluted or adding artificial constraints designed to intentionally frustrate competitors. We believe this practice is detrimental to the utility of the competition as very few teams will solve these challenges (with the main differentiating factor being luck). Two particularly memorable examples of this are from Ghost in the Shellcode (GitS) 2013 and 2014. HackerBook (GitS 2013) asked competitors to identify famous and lesser-known hackers from their photographs, which was difficult given that competitors were provided with no information aside from a picture, and not every picture was available on the Internet. Revenge of the Imgception (GitS 2014) asked competitors to forensically analyze images in order to find a hidden key. The challenge's ambiguity provided too many potential paths that could be followed by a competitor and made it too difficult to be solved by any competitor during the competition.

Occasionally, there are challenges that are designed so poorly that they are not solved during the length of the competition due to ambiguity or difficulty. These challenges just stress competitors unnecessarily and we have seen even experienced competitors give up after failing to succeed at a challenge. It's important to note that some competition organizers will deliberately develop a challenge specifically designed to stress their competitors, in which case this the idea that the challenge is designed poorly is irrelevant.

Another common means of making a challenge harder is by adding a brute-force component, requiring participants to rely on luck or the passage of time to find a solution. Brute-force is, however, not a meaningful learning objective in the context of a CTF, where gameplay and time-management are a concern. Achieving good gameplay with any challenge is difficult. The distinction between a challenge that can be solved in a reasonable amount of time and one that cannot is often a very fine line. Challenges that require brute-force almost always cross this line, and as such should usually be avoided. One remedy is to make obvious that the challenge requires brute-force and to keep the key space is small.

Common problems involving brute-force are guessing a password, a file name or location, or guessing a randomized address. An example of a challenge foisting brute-force on a competitor is Misc 300 (DEFKTHON CTF). This challenge provided recursively password protected zip files; in essence, each password protected zip held another password protected zip. After writing a script that would recursively crack each zip, a final zip with a harder password must be cracked. After finally getting to the actual challenge, a sound file, the challenger is expected to open the audio file in Audacity and click view as a spectrogram to reveal the key [8]. This challenge obfuscates any potential learning objectives and requires significant guessing. In addition. this challenge provides no real suggestion that the sound file should be seen as a spectrogram. This challenge highlights many issues in current CTF challenge design and the ambiguity and difficulty that arise from that.

Conversely, challenges that are easily solved are not necessarily detrimental to the competition but their quantity must be kept low in order to achieve a bal-

anced and enjoyable competition. Very often CTF competitions make use of number "booster" challenges that have very easy answers. Examples include CSAW

```
#!/usr/bin/env bash

while [ -e *.zip ]; do
 files=*.zip;
 for file in $files; do
   echo -n "Cracking ${file}… ";
   output="$(fcrackzip -u -l 1-6 -c '1' *.zip | tr -d
'\n')";
   password="${output/PASSWORD FOUND\!\!\!\!: pw == /}";
   if [ -z "${password}" ]; then
     echo "Failed to find password";
     break 2;
   fi;
   echo "Found password: \`${password}\`";
   unzip -q -P "${password}" "$file";
   rm "${file}";
 done;
done;
```

**Figure 2: Brute-force solver for a portion of the DEFKTHON Misc 300 challenge [8]**

CTF's Trivia category or PlaidCTF 2014's Sanity Check challenge. These challenges serve to boost participation numbers as most CTFs only count competitors that have solved a challenge in their statistics. An exception is the "Hack The Planet" challenge made popular by DEFCON [1]. This challenge is a CTF trope due to its inclusion in multiple DEFCON CTFs. Its inclusion in other CTFs provides newcomers a glimpse into the culture of computer security professionals.

A well-designed challenge should lead the competitor through its own solution process. This means that the challenge inherently provides digital breadcrumbs that can be used by a frustrated competitor to move toward the final solution. A memorable example is Jordan Wiens' Recon challenge (CSAW 2013). The challenge provided was "Michael Vario sure does some suspicious signs, hope he doesn't do me." The competitor can infer that someone named Michael Vario is now involved. Researching Michael Vario tells you that he works with PGP. The intended logical conclusion is for the competitor to find Jordan's public PGP key. Searching on public key servers will reveal a PGP key that claims "Getting warmer" immediately verifying that the competitor is are on the right track. Finally opening the PGP key's embedded image will reveal the answer to the challenge. The challenge completely steps the team through its own solution process, which gives valuable feedback to the competitor and hints as they may need them.

One of the most important forms of feedback is letting the competitor know when they have found a key. If the competitor is unable to identify that they've acquired the key, they may have wasted hours trying to solve the challenge when in reality they've already solved it. Many CTF competitions use a key format that is helpful for identifying keys. CSAW CTF's key format which has been used by other CTF's is: key{flag} where flag is what should be submitted to the scoreboard.

**2.3. Quality Assurance**

For most challenges, it should be obvious to veterans as to what to do to solve it. But with newcomers, it is important that they provided in-game guidance toward the correct answer. Newcomers often will need to be introduced to new topics in security. To help catalyze this learning experience, the challenge should provide some introduction in some manner similar to Jordan Wiens' Recon challenge (CSAW 2013), as described above.

It's not necessary to completely lead the competitor, as this would remove the benefits of the self-taught nature of CTF competitions. In CSAW CTF, the challenge developers are well-known, industry experts and the organizer is a student at NYU Engineering. Having a student organizer that is a near-peer to the target audience provides a valuable perspective to challenge developer. The student is typically very understanding of how a student will approach the problem, can identify scenarios where the challenge may be too ambiguous, and work with the developer on improving it. In our experience, this combination improves the quality of the challenges and competition that promotes a better learning environment.

After a challenge has been developed, the quality assurance (QA) phase begins. For CSAW CTF the QA phase consists of other judges and the competition lead attempting to solve the challenge in different ways, considering how those with less knowledge would approach solving it. Other judges review the challenges, notes are taken, and the challenge goes through another iteration of development. This gives the challenge developers another perspective that allows them to improve upon the challenge agrees on a fair point value.

This approach to QA is similar to other CTF competitions such as Ghost in the Shellcode. Sadly, while the QA phase is arguably the most important of the challenge development lifecycle, it is frequently skipped in most CTF competitions. Some competitions are much less involved with QA and a common scenario is the challenge organizer releasing challenges a few minutes after they've been written. This lack of QA very often leads to challenges whose point values do not reflect the difficulty of the challenge, unsolvable challenges, and improperly configured challenges. We have learned this lesson through experience. For example, SCP-hack (CSAW CTF 2013), which was valued at 500 points, the typical maximum valuation for a challenge, was not solvable for a period of time due to it operating behind

a closed port. In addition, the challenge itself involved requiring the competitors to perform difficult tasks in order to even begin the challenge (*i.e.* connect to the server from multiple countries). Similarly, Noobs First Firmware Mod (CSAW CTF 2013) was released with an incorrect key, which lead to confusion and multiple challenge revisions in the middle of the competition. Both of these scenarios could have been prevented with better and more precise QA.

Another high profile example occurred during the DEFCON 21 CTF Finals in which flags were not properly attributed back to the teams that had earned them. An improperly configured network prevented certain teams from being able to redeem their keys that affected the scores during the competition but was rectified post-competition. [9]

In an optimal QA setting, each challenge would be solved by a party with no prior knowledge of the challenge. The solution generated would be used to validate the challenge is both solvable and properly configured. Due to this rarely being the case, we suggest that challenge developers create solvers for their own challenge to aid in the QA phase and make it simpler to test challenges during a live CTF competition. A great example of this is Silk Street (CSAW CTF 2013 Finals). The challenge came fully prepared with a description, hints, and a challenge solver from the developer. This helped other judges validate the challenge performed according to specifications and also allowed organizers to quickly test the challenge during the competition.

### 2.4. Challenge Point Value and Adverse Effects on Competitors and Organizers

After generating a point value appropriate for the challenge (by using the QA process or not) and the challenge is made available online, an interesting phenomenon is observed by organizers. Competitors will immediately and superficially assess the difficulty of a challenge prior to any actual knowledge about the challenge based solely on its point value–challenge point values represent the first impression of the challenge to the competitor. Although competitors should not judge a challenge by its point value alone, competition organizers see this happen all the time by noticing the differences in how many teams solve a challenge despite similar difficulties but similar point values. Organizers can only guess that high value challenges convince low-spirited participants that they're not solvable with their current skill set. While organizers cannot completely avoid these misperceptions, perhaps they can properly value challenges so that they convey the right impression and level of knowledge using other multimodal valuation techniques. Challenge avoidance is essentially an unsolved problem in CTF participation.

In CSAW CTF, challenges are valued by a single entity, the competition organizer, with input from the challenge developer. This reduces subjectivity and allows for a straightforward increment in points and difficulty. As the competition organizer is supposed to be experienced both with CTFs and student mentality, he or she can determine which challenges will be more difficult for a student and which ones will be easier. This introduces a bias inherent to the organizer as the organizer will prefer certain types of problems but this kind of bias is part of every CTF. For example, DEFCON prefers reverse engineering and exploitation problems, whereas other CTFs value web challenges.

### 2.5. Challenge Infrastructure

The competition website is typically how competitors will interface with your scoring and key submission services. This makes the competition website one of the most important aspects of the competition [2]. Broken websites are a common problem in CTF competitions. CTF sites have failed due to load, a failure to prevent key brute forcing, a failure to load challenges, and even allowing competitors to score points arbitrarily. It's important to properly test challenge infrastructure so that it is not accessed early, keys can't be brute-forced during the competition, and challengers cannot score points arbitrarily or access each other accounts or private challenge data [10].

CSAW CTF prevents brute-force by using a key submission form specific to each challenge (as opposed to a universal form) and uses rate-limited key submission interface to further prevent teams from brute-forcing keys. While something of a nuisance to players and developers, and as such, not implemented very often, it is absolutely important to implement a rate limit as it almost entirely eliminates the brute-force issue in terms of key submission.

The challenge infrastructure should be developed such that teams can be attributed to IP addresses, logs should be collected, and actions can be held attributable to a team. Cheating is a problem in CTF competitions, just like any other type of competition, and the infrastructure should do its best to prevent and allow detection of cheating. The challenge infrastructure must be built defensively incorporating all the standards of web security, where applicable. It is not unheard of for competitors to begin attack both the challenge website and other teams through the website.

Before release, challenge infrastructure should undergo a code audit by multiple individuals so that as many bugs can be caught before the competition as possible. This is an arduous process, requiring an organizer to adamantly monitoring site logs during testing. The infrastructure hosting the challenges themselves should also undergo review. A common scenario is a server that is improperly configured and the challenge key file is changed so that other competitors cannot submit the key. This type of unsportsmanlike conduct should always be protected against. Setting proper permissions on applicable files and mounting the challenge server's disks read-only are appropriate solutions [10].

#### 2.6. Known Unknowns

Despite efforts to maintain an appropriate point rating, an issue arises when both the competition organizer and challenge developer misunderstand the difficulty and ambiguousness of a challenge. A good example of this issue is Miscellaneous 200 (CSAW CTF 2013). This challenge presented competitors with a PNG image modified to display at a smaller size than its actual size, thus hiding information and the key. Both the challenge developer and competition organizer failed to understand the ambiguity of the challenge and how competitors would approach the challenge. This resulted in a drastic drop in solves for the challenge compared to other challenges in the same category and many competitor complaints during and after the competition. Both the competition organizer and challenge developer incorrectly believed that the solution path they had chosen would be the most common.

Aside from points and QA, the challenge's description and hints that are released alongside the challenge are important and should be considered carefully. Very often the challenge description is completely omitted or given very little thought. Challenges should come with a story or theme that provides glue between the challenges so as to help keep competitors involved in the competition, as they were in PicoCTF 2013. The description can also be used as a mechanic to be used in the actual challenge providing email addresses, hidden hints, or even jokes. Challenge solution hints can also be used for the same purpose but the key difference between hints and descriptions is that with a hint it is difficult to place exactly where a user is in the challenge. The released hint may not be applicable to many competitors if any. In many different competitions hints are often created before the competition begins and also during the competition to help make hints more relevant.

## 3. Conclusions

The CTF model is an area of great potential which is still open to refinement. CTF competitions as a learning and technical evaluation platform are quickly becoming more popular. CTF competitions can be used for many different applications including competitions, education, and even interviews. It has been very successful for learning and evaluation and has been deployed with much success in many different environments. DEFCON CTF is the standard by which the industry measures the top security teams in the world and competitions like CSAW CTF and PicoCTF are regarded as excellent introductions to both CTF and the security community.

However, the community cannot continually acknowledge CTF as an infallible resource for education for everyone, every time. CTF is a partnership between the organizers and competitors and the community must recognize that it is difficult to organize and compete in CTF competitions. Organizers are not perfect and players are not guaranteed to benefit by playing.

## 4. References


[1] psifertex, "DEFCON 17: Maximum CTF: Getting the Most Out of Capture the Flag," [Online]. Available: https://www.youtube.com/watch?v=okPWY0FeUoU.

[2] fuzyll and psifertex, "The Many Maxims of Maximally Effective CTFs," [Online]. Available: http://captf.com/maxims.html.

[3] atlas, "DEFCON 14: The Making of atlas: Kiddie to Hacker in 5 Sleepless Nights," [Online]. Available: http://www.youtube.com/watch?v=gYOy7CGpPIU.

[4] C. Eagle, "Organizing and Participating in Computer Network Attack and Defense Exercises," [Online]. Available: http://www.nps.edu/video/portal/Video.aspx?enc=Fvcj9jTKwtwcxg2Wgv3NOEGEdfe6jktD.

[5] M. Arpaia and T. Reed, "Beat to 1337: Create a successful university cyber defense organization," [Online]. Available: https://www.youtube.com/watch?v=Xe6y-mOOVX0.

[6] E. N. Team, "Intro to CTF," [Online]. Available: http://bambuser.com/v/1989728.

[7] "29C3 CTF Exploitation 100 Minesweeper," [Online]. Available: http://www.blue-lotus.net/29c3-ctf-minesweeper/.

[8] mathiasbynens, "DEFKTHON CTF: Miscellaneous 300," [Online]. Available: https://github.com/ctfs/write-ups/tree/master/defkthon-ctf/misc-300.

[9] V. Genovese, "Final Writeup," [Online]. Available: https://blog.legitbs.net/2013/08/final-writeup.html.

[10] ryan0rz, "What does the infrastructure of various CTFs look like?," [Online]. Available: http://www.reddit.com/r/securityCTF/comments/1ntoue/what_does_the_infrastructure_of_various_ctfs_look/ccm4bt5.

[11] tylerni7, “Why CTF” [Online]. Available: http://ppp.cylab.cmu.edu/wordpress/?p=1182