Journal of Cybersecurity Education, Research and Practice

Volume 2022 | Number 2 Article 9

January 2023

# Teaching by Practice: Shaping Secure Coding Mentalities through Cybersecurity CTFs

Jazmin Collins
*Arcadia University*, collinsjazmin.21@gmail.com

Vitaly Ford
*Arcadia University*, fordv@arcadia.edu

# Teaching by Practice: Shaping Secure Coding Mentalities through Cybersecurity CTFs


## Abstract

The use of the Capture the Flag (CTF)-style competitions has grown popular in a variety of environments as a method to improve or reinforce cybersecurity techniques. However, while these competitions have shown promise in student engagement, enjoyment, and the teaching of essential workforce cybersecurity concepts, many of these CTF challenges have largely focused on cybersecurity as a general topic. Further, most in-school CTF challenges are designed with technical institutes in mind, prepping only experienced or upper-level students in cybersecurity studies for real-world challenges. Our paper aims to focus on the setting of a liberal arts institute, emphasizing secure coding as the focus of CTF-engaged learning for beginner to upper-level undergraduate students. We propose a survey system to evaluate the secure coding mentality of our students before and after taking these challenges, as well as an easily-hosted, low-resource CTF platform that students can access either in or outside of the classroom. We have found this system to be moderately effective at framing and improving the secure coding mentalities of our students.



## Keywords

Cybersecurity Education, CTF, Secure Coding

## Cover Page Footnote

We thank the Computer Science and Mathematics department of Arcadia University for their support during this project. We are eternally grateful for the contributions made by Amela Gjishti (testing and support), Konstantin Menako (PHP challenges), Alexandr Chebatarev (deployment automation, C++/JS challenges), and Daniel Tyler (initialization of the dockerization process).

# INTRODUCTION

With the rapidly growing scope of technology, there has been a consequently growing demand for cybersecurity professionals from nearly all sectors of industry. Secure practices in coding and code evaluation are a crucial part of creating a safe digital world, but unfortunately, many software engineers and programmers entering the workforce lack experience with such practices (Paulsen et al., 2012). This is largely due to the struggles faced by current cybersecurity educators, who face an increasingly intense demand for cybersecurity instruction, without an increased flow of resources that they can use to support this instruction (Cobb, 2016).

In particular, we have noticed that cybersecurity educators are often tasked with preparing their students for both theoretical and practical aspects of cybersecurity, with greater challenges presented for the practical side. The theories within cybersecurity education are generally simpler to introduce and create traditional, lecture-based curriculum for, and there are a fair amount of resources present for educators to utilize in creating their theoretical curriculum. Some examples of these resources include online articles, information databases, or conferences sponsored by government cybersecurity education initiatives, such as NICE (https://www.nist.gov/itl/applied-cybersecurity/nice) and CISA (https://www.cisa.gov/cybersecurity-education-career-development). These programs give educators access to shared resources that they can introduce into their classrooms, and provide useful hubs of information explaining theoretical concepts for cybersecurity.

However, on the practical side of things, educators' access to useful resources is much more limited. Generally, educators require platforms that they can utilize in class to give their students hands-on experience. These platforms can be expensive to host, complicated to set-up or scale to match the size of a class, and difficult to maintain for an educator already balancing their other responsibilities as an instructor. There are some programs that seek to offer educators pre-prepared spaces where they can imitate practical, workforce scenarios for their students, such as the NICE Challenge (https://nice-challenge.com/), but there is not an abundance of such practicality-focused platforms. Further, we have noted that these resources - as well as the theoretical resources discussed above - tend to focus on broader defense-related topics in security, with a noticeable lack of attention to other sides of cybersecurity. In particular, there is rarely a focus on secure coding as a training method for cybersecurity professionals, or offensive tactics of security evaluation.

Our project seeks to add to the practical resources that cybersecurity educators can utilize in their courses to increase and improve the secure coding mentality of their students. We specifically plan to create resources that highlight secure coding and offensive security practices through the common cybersecurity competition

tactic known as capture-the-flag (CTF) challenges. The reasons for this specific goal and the classroom-based framework within which we set up our challenges will be discussed at further length in later sections.

# MOTIVATION

There is much that needs to be improved within the field of cybersecurity education, specifically in teaching secure coding practices and mindsets. Many CS undergraduate students begin to learn the foundations of their coding skills early in their education, without much thought or preparation given to secure practices while coding. This leads to an incoming software engineering workforce unused to thinking of the security of the code they produce, and a belated need to check for security exploits and attack vectors after software is already in production (Gasiba et al., 2020; Taylor and Kaza, 2011; Chi et al., 2013). Unless students are taught to be security-conscious while coding, employing secure coding practices as they write software instead of afterwards, this will continue to be a costly, common problem in the tech industry.

The primary objective of this project was to employ CTF coding challenges as a means to improve the consideration of secure coding among undergraduate students who are interested in programming. Subsequently, the major learning objective is for the students to change their coding mindset by considering security as a vital step in the development process rather than keeping it as a mere afterthought. Another core learning objective is for the students to get a feel of what to look for in their code when they conduct its security evaluation. We would test how students approach code before the CTF challenges and how they approached code after attempting them, to see if their mentality in checking the security of the code changed at all. Ideally, we wanted to increase the secure coding mindsets of the undergraduate students so that they would begin to write and evaluate code with secure practices in mind. In doing so, we hope to help produce software engineers more equipped for the necessity of security-conscious programming in the growing tech industry. Our other objectives with this project were that:

- Students should have an increased practical understanding of web security. They should also have an increased understanding of typical attack vectors in web security, and be able to more easily comprehend the attacking of insecure code. We hoped that this would represent increased confidence in topics of web security, and a recognition of the importance of building web applications in particular with secure coding practices in mind.

- Our CTF challenges would influence increased engagement with secure coding concepts, as well as improved interest in the cybersecurity field at our university. Coming from the background of a liberal arts university, with a smaller CS program, students primarily focus on software development and rarely take cybersecurity electives. We hope that secure coding concepts would draw more interest in cybersecurity among undergraduates.
- Students should have access to a variety of CTF-related and secure coding-related resources following the completion of the course that they feel comfortable using. This will allow them to pursue their interest in cybersecurity outside of the class and build up a steady project base in their own time. Ideally, this will allow us to spread the importance of secure coding outside of the course content itself, and continue to encourage students to think about and prioritize it even once the CTF challenges are over. It will also encourage independent learning and engagement with cybersecurity in our university community.

We have designed the framework and resources of our project with these objectives in mind, establishing a customized CTF challenge educational platform that will be expounded upon in the following section. We incorporated the CTF challenges into the Introduction to Network Security course that does not have any prerequisites and is structured to be taken by non-CS majors. It is important to note that the above-mentioned objectives are not related to the learning objectives in the network security class, with the idea that the CTF challenges can be introduced in any CS course (where it is deemed appropriate).

# RELATED WORK

The use of CTF-style competitions has grown popular in a variety of environments as a method to improve or reinforce cybersecurity techniques. Several previous papers have discussed the usage of CTF as a gamified method of enhancing cybersecurity performance for individuals in the technology fields (Taylor et al., 2013). Generally, there appears to have been the greatest amount of focus on traditional, competition CTF challenges, hosted in city-wide or even nation-wide events where teams compete against one another in a variety of CTF styles. Gasiba et. al. have utilized this setting, usually thought of as a space for undergraduate student engagement, as a training setting for industry cybersecurity professionals (Gasiba et al., 2020). Their work demonstrates the need for increased cybersecurity excellence continuing even with current industry workers. They bring to light the value of CTFs as a means to reinforce or teach concepts often overlooked by workers in a time-constrained, intense competition. They affirm the benefits of using CTF in education settings, and

believe that CTF should be used for training in the industry as well as academia. The competition-style CTF has also been used as a tactic to encourage undergraduate student interest in cybersecurity, with studies by Davis et al. demonstrating their use as a recruitment tool for students interested in cybersecurity and an encouragement to increase student knowledge of their own cybersecurity concepts outside of the classroom. Specifically, Davis et al. designed apps allowing students to launch their own CTF for practice at home. Davis et. al. also adds within their own study that while the CTF competitions can work to encourage students to learn more on their own, it does not seem effective as its own education tool (Davis et al., 2014).

However, there are other settings where CTF has been used specifically as a tool for education. The tactic of integrating small modules of CTF into the course curriculum is introduced by Mirkovic and Peterson, who developed CTF challenges to give to students in an undergraduate security class. Mirkovic and Peterson's challenges were introduced 2-3 times a semester, allowing students a few weeks in between to prepare for the challenges and treating them like a competition-style CTF. Their class CTFs have a slightly lighter load than competition CTFs, though, making them easier to implement (Mirkovic and Peterson, 2014). Chothia and Novakovic also implemented in-class CTF challenges in a separate study, where CTF challenges were designed as the primary part of a cybersecurity course's classwork throughout an 11-week semester. The course was set up with a student expectation of 3-4 hours of work on CTF challenges, along with a write-up describing how each flag was obtained. Programs were designed to be completed on VMs offline (Chothia and Novakovic, 2015). These integrations primarily take the CTFs as modules within the coursework, designed to be completed by students as they learn, as a manner of using gamification (Gonzalez et al., 2017) to encourage participation and active learning/retention of cybersecurity concepts. Gonzales et. al. takes a slightly different approach with their use of CTF in the academic setting, placing the challenge instead at the end of the school semester and utilizing it primarily as a reinforcement technique for what students have been taught by the ordinary class curriculum in a cybersecurity course (Gonzalez et al., 2019). Success in student engagement with the core concepts of cybersecurity classes has generally been met through all versions of the academia CTF challenges, with high participation and reported enjoyment from students for the challenges themselves.

For all academic settings, a constant challenge for incorporating CTFs is the creation of enough challenges to satisfy all students. A study by Schreuders and Ardern investigated a technique of using machine learning to generate random challenges for students, but ran into difficulties with students recognizing the same general patterns after a while and being able to more quickly complete the tasks (Schreuders and Ardern, 2015). Another consistent challenge for academia CTFs is

the manner of testing how effective the CTFs are at teaching or reinforcing concepts for students. A variety of evaluation methods are presented, and a particularly thorough one by Chothia and Nvakovic utilizes open-ended student feedback as well as point systems to determine material learned. A correlation was shown between the understanding of basic cybersecurity concepts greatly strengthened by the number of flags achieved, but the correlation was weaker for the students with higher marks, and it couldn't indicate a deeper understanding of the concepts. Describing fixes for the problems helped demonstrate which students had a deeper understanding. This study, however, admits the pitfalls that even with a thorough evaluation technique, it becomes difficult at the higher level to tell how deeply students are understanding and retaining concepts based purely on the measurements in place (Chothia and Novakovic, 2015).

# SYSTEM OVERVIEW

## General Framework

Our CTF challenges (open-source, dockerized, available on GitHub, and publicly-deployed on Oracle Cloud always-free servers for everyone to use at (Ford, 2022a, Ford, 2022b)) are set up on the main CTFd server (Chung, 2017) being run 24/7. CTFd is an open-source server for running CTF challenges which we have utilized as the base of our project for its easy set-up and website design. CTFd does not have any challenges on its own. Our challenges are designed in the Jeopardy-style of CTF, where students will form teams to find exploits and attack vectors of various hosted webpages, and uncover the flags (strings of text and numbers) hidden within these vulnerabilities. We developed a total of 20 challenges, inspired by real-world vulnerabilities that we searched for on Vulmon (Vulmon, 2022), for instance, Heartbleed Vulnerability (Vulmon, 2014) and SQL Truncation Vulnerability (Vulmon, 2008). We simplified the exploitation chain for those vulnerabilities, making it feasible to approach by people who have never worked on CTFs or thought about secure coding at all. By submitting flags to the main CTFd server, students then gain points for their team. These points are used in ranking the teams against one another on a scoreboard, and also in determining extra credit received in the main course the CTF competition is being held within (see section Integration with the Course for more details).

Our CTF challenges are written in a variety of languages, encouraging students to interact with various programming languages they may encounter in security fields and situations later in the industry. The highest number of challenges are written in PHP or a combination of PHP and Javascript, giving priority to languages commonly used in website design and web security. Challenges written in Java are the second

highest in number, followed by a handful of challenges in Python that are the third highest. One challenge is written in C++. Each of these challenges is also set up with downloadable files of source code which students can analyze while they are trying to find the flag (see section Viewing Source Code for more details). These files of source code are written in the languages each challenge is marked as (for instance, PHP or PHP/Javascript, Java, and so on).

As students progress through capturing flags, the challenges are designed to increase in difficulty, with the exact vulnerabilities present in each challenge webpage going beyond what is taught in the class. Students will be highly encouraged to look things up and utilize the Internet to figure out what they need to do next, mirroring the traditional CTF-style competition setting, instead of asking the professor for assistance. Teams will also be able to "purchase" hints through the CTFd hint system, with a varying number of hints available on each challenge. These hints will cost points from the teams' overall scores, again to encourage students to discover the answers on their own before buying them. However, if a team of students finds themselves completely lost, even with the assistance of hints and the Internet, they may ask the professor for guidance in solving certain challenges. This will be treated only as a last-resort option, though, to keep in the spirit of the CTF competitions and as a result of these CTFs being treated as an extra credit assignment for class (again, see section Integration with the Course for more details).

## Viewing Source Code

As part of the CTF challenges, students will be able to download source code files corresponding to each challenge webpage. These files are designed to be examined while students test out site vulnerabilities on the webpage, and serve as a guide for students to see what parts of the website code might be vulnerable to attack. This approach to evaluating the system mimics white-box cryptography (Saxena et al., 2009; Beunardeau et al., 2016), and allows students to spend less time figuring out what is inside the system they are trying to exploit, and more time discovering the problems within it straight from the beginning. By reading through the source code, students can look for vulnerabilities written into the code, and try exploiting them on the corresponding website as though they are system evaluators rather than attackers.

The idea for this white-box approach to our CTF challenges rather than the typical black box comes from work with encryption algorithms. Effective encryption algorithms are public and open-source, allowing them to be reviewed heavily by any parties interested in reviewing them for vulnerabilities. The security of these algorithms is not dependent on an intimate knowledge of the system they come from, but rather on the evaluation of how they are written and what can be done to

improve them. Instead of hiding vulnerabilities, they can be openly available for anyone to find and fix, leading to a less "dangerous" environment, as Bellovin and Bush call it (Bellovin and Bush, 2002). Encryption algorithms and schemes are often public rather than proprietary due to this largely beneficial state of community involvement and improvement, rebuking the concept of "security through obscurity" (Edwards, 2014) that many proprietary systems follow, and sometimes suffer due to. By mirroring the white-box encryption algorithm approach, our students are able to become invested reviewers of the security of our challenge webpages, able to test its weaknesses and identify the problematic elements of code that need to be changed within it through familiarity with security practices.

Our source code approach also has other ramifications for our students, and in particular, for our main objective with this project. As we are trying to increase the secure coding mentality of our students, encouraging them to think of security problems as they code instead of afterwards, giving them access to source code itself is invaluable. It allows them to start learning how to read code with security in mind, and how to think of ways to look for code vulnerabilities as they write their own programs. They begin to think of how to interact with code securely rather than simply how to interact with an exploitable website, building the secure mentality we are looking for.

Additionally, by having the source code, students from a variety of coding backgrounds are able to read and understand the weaknesses present in every challenge even without a "proper" background in each coding language. Since they have direct access to source code, they can use their knowledge of whichever coding languages they are most familiar with to parse through unknown coding languages, looking up language-specific terms where needed, but otherwise using simply their own general knowledge. They begin to learn to figure things out in unknown languages with a coding mindset, and develop methods of dealing with problems cross-language through a programming mentality. By making source code accessible for our students, we also remove the barriers that typically divert beginners away from participating in CTFs.

## Integration with the Course

Our CTF competition is embedded within the framework of an undergraduate network security class, taken as an elective within our university's CS degree curriculum by students varying in background from freshmen to seniors. The course does not assume any prior knowledge in any of the CS areas. The CTF competition is designed to serve as a supplement to the topics being discussed within the network security class and opened for two months in the middle of the semester. It is meant to give a

practical and hands-on look at the topics outside of the class (the challenges' topics are not directly covered in the class), and while not included as a required part of the curriculum, it will be used for extra credit (bonus points) for students who participate in it. Participating students will organize themselves into teams of two, reflective of common pair programming techniques in CS education.

It is important to note that even though the CTF challenges are used in a network security class at our institution, they are developed to be easily integrated into any CS course where it makes sense to introduce practical secure coding. Also, students who have never programmed before would struggle to work on these challenges. However, in our particular case, first-year students were still able to solve Java challenges as they have taken their first Java course in the fall, and the network security class runs in the spring. In fact, most of the students in the class were only familiar with Java but that did not hinder them from solving challenges written in Python or PHP due to the hints, comments, and resources integrated into the CTF challenges.

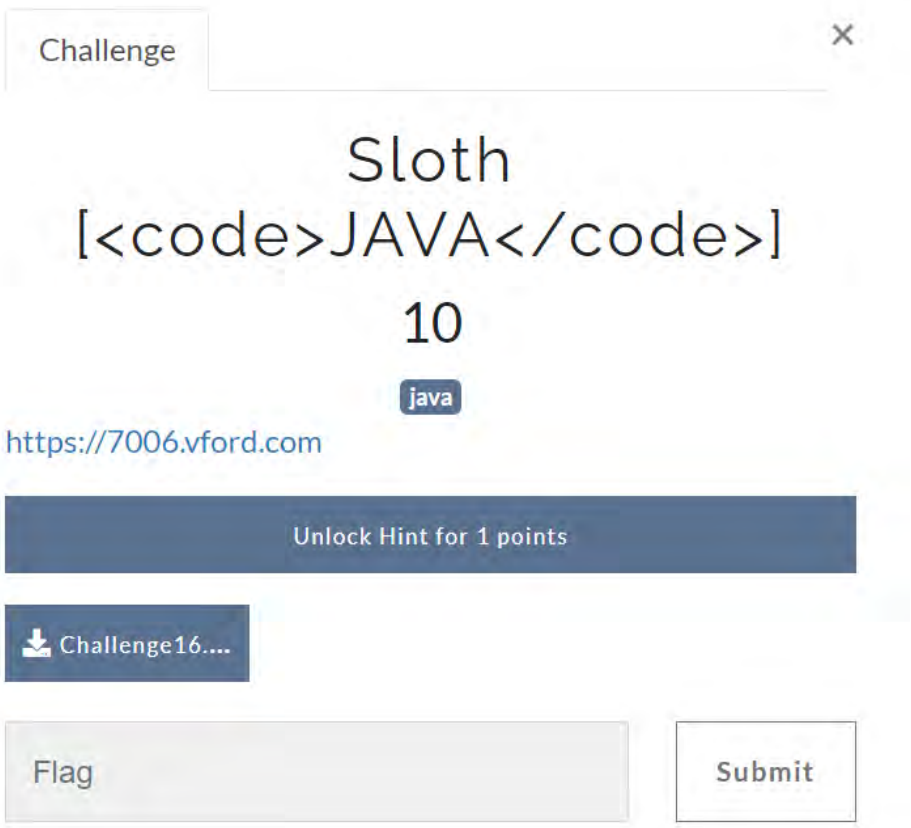
Depending on the number of challenges teams of students are able to complete, and thus the amounts of points they receive, students will be given a number of additional points added to their main tests in the class. Specifically, for every challenge they are able to solve, ten points will be added to their test grades, starting with the midterm and then the final. Full credit on both tests (each worth 5% of the final course grade) is feasible with this point system, if 20 challenges are solved. The course is explicitly designed so that not participating will not impact a student's grade in any way, though significant advantages can be gained by completing the challenges. The challenges will be continuously running as soon as they are launched, and students are given some opportunities to work on the challenges in class, though they may also continue to work on them from home. The amount of time in which they devote to the challenges is entirely up to them.

The CTF competition is launched in March during the Spring semester of the security class, after students have consumed a large portion of the class content. Not all students within the network security class are expected to have coding experience, as it is a course open to beginners in the CS fields without prerequisites and sometimes is taken by non-CS majors. Thus, it is primarily students with coding experience within the class who want to take their practical skills a step further that participate in the challenges, though all students are highly encouraged to participate in the CTF challenges after they are launched.

## Challenge Examples

When students begin to approach these challenges, they first select one they wish to pursue from the main CTFd server home page. After selecting one, a screen will pop

*Figure 1: Image of the general pop-up that displays for a Java CTF Challenge called "Sloth"*




up to give students a link to the webpage they will be trying to exploit, download options for any source code files they will need for the challenge, and an option to unlock hints. There will also be an area to submit the flag text as soon as it is found (see figure 1).

From this screen, students can open the webpage they will be looking at and begin to examine the code behind it, trying to find vulnerabilities in the webpage. For many of the earlier challenges, the source code will be commented with subtle hints or messages from a "developer" that may give them clues as to where they need to direct their attention. In the Java program shown in figure 2 for instance, a comment is left above a hash saying that the hash is "just for testing purposes, please change when deploying in production!!". While not all comment hints are this obvious, lines

such as this may draw students' attention to suspicious or vulnerable functions in the code that they need to pay attention to.

*Figure 2: Image of part of the Java source code within the "Sloth" CTF challenge, displaying developer comments*

```
public static void main(String[] args) throws Exception {
    // hashing the password helps security
    // this hash corresponds to one of the commonly used passwords in 2018
    // just for testing purposes, please change when deploying in production!!!
    adminPasswordHash = "0571749e2ac330a7455809c6b0e7af90";

    // read the challenge file and store it in the data
    File file = new File("flag16.txt");
    BufferedReader br = new BufferedReader(new FileReader(file));
    String line = "";
    while ((line = br.readLine()) != null) data += line + "<br>";
    br.close();
```

Students within the example problem may realize the next steps they should experiment with thanks to such hints, such as finding a de-hashing website and running the hash code through it along with the content of the variable password. If they manage this, they will find the correct credentials and be able to log into the exploitable webpage (see figure 3). For this problem, after a student logs in, they will next be able to access the flag file and submit it.

Not all exploitable webpages within the CTF challenges will look or act the same. Another example of a CTF challenge students might come across is a Python challenge called "Da Bug", which contains the remnants of debugging commands that can be used insecurely by users. By noticing the comments above this section of the code, and realizing that admin commands in Linux are still available to anyone who enters the website, students may begin to examine the commands available to them in Linux and see which ones are necessary to retrieve a flag file. The tricky portion of this challenge would be figuring out where to interact with the webpage to put the commands, as there are no input boxes on this website, such as with the password fields from before (see figure 4). Students may figure out that the URL path is another interactive portion of the website, and that Python code permits users to enter Linux commands within it. They may discover this by purchasing hints, recalling class lessons, or playing around with the website. From here, it is a short journey for students to enter the correct Linux commands necessary to access the flag file and submit it for points.

*Figure 3: Image of the exploitable webpage for the Java CTF challenge, "Sloth"*


The two challenges covered above are both beginner-level challenges that would come early in the CTF competition stages. However, other challenges work also very similar to the process described above, with students having access to the webpage and source code, and looking between the two of them to see which portions are vulnerable. Often, the hints within comments on the code - which range from the more obvious notes given above to a simple line describing a function of a piece of code that may be useful for a student's attack attempts - will be the first step in guiding students to the right answer. These comments are especially helpful for the beginner population of our network security class, as many of them have no experience with cybersecurity before this and would not know what to look for.

# RESULTS

Prior to the beginning of the challenges, our participating students completed a pre-survey indicating their level of agreement (via the Likert scale) with a variety of statements. These statements were designed to reflect the core objectives of our project, gauging the students' pre-challenge secure coding mindsets, understanding of web security, interest in cybersecurity, and access to cybersecurity resources. The same questions were provided as a post-survey following the challenge period, with additional short response questions as well as a final open-ended question asking for their thoughts on the challenges. The data that we collected from both surveys, and specifically, the overall changes we are able to see in the class mindset following the

*Figure 4: Image of the exploitable webpage for the Python CTF challenge, "Da Bug"*


completion of the challenges, have been examined to evaluate the results of our CTF challenges.

## Pre-Survey Results

We received a total of 28 responses from our student participants on the pre-survey of five, Likert-style questions. Although we had 28 submissions to the form, only 19 participants later registered as active users on the site to complete the secure coding challenges.

The five statements that our students were asked to rate their agreement with during the pre-survey are the following:

- I often think of evaluating my code for security when I write or see it.
- I can comfortably analyze my code using attack and/or vulnerability vectors.
- While navigating the Internet, I have a high level of awareness of web attacks and/or web attack vectors.
- I am currently interested in topics of cybersecurity and secure coding.
- I have a reliable amount of cybersecurity/secure coding resources that I can use often and easily.

Of these five questions, the first two are designed to evaluate the secure coding mindset, the third evaluates understanding of web security, the fourth evaluates

cybersecurity interests, and the fifth evaluates access to cybersecurity resources. All of our questions evaluate these through the students' self-reported assessments, due to the goals of this study as well as the course our students were in. Since this course was an elective introduction to network security, the goal was not to make all students proficient in identifying bugs in code. Similarly, the goal of our study was not security proficiency, either. It was to see if students noticed changes in their mindsets and the way they approach secure coding and coding in general, which we decided would be best noted through the perspective questions listed above.

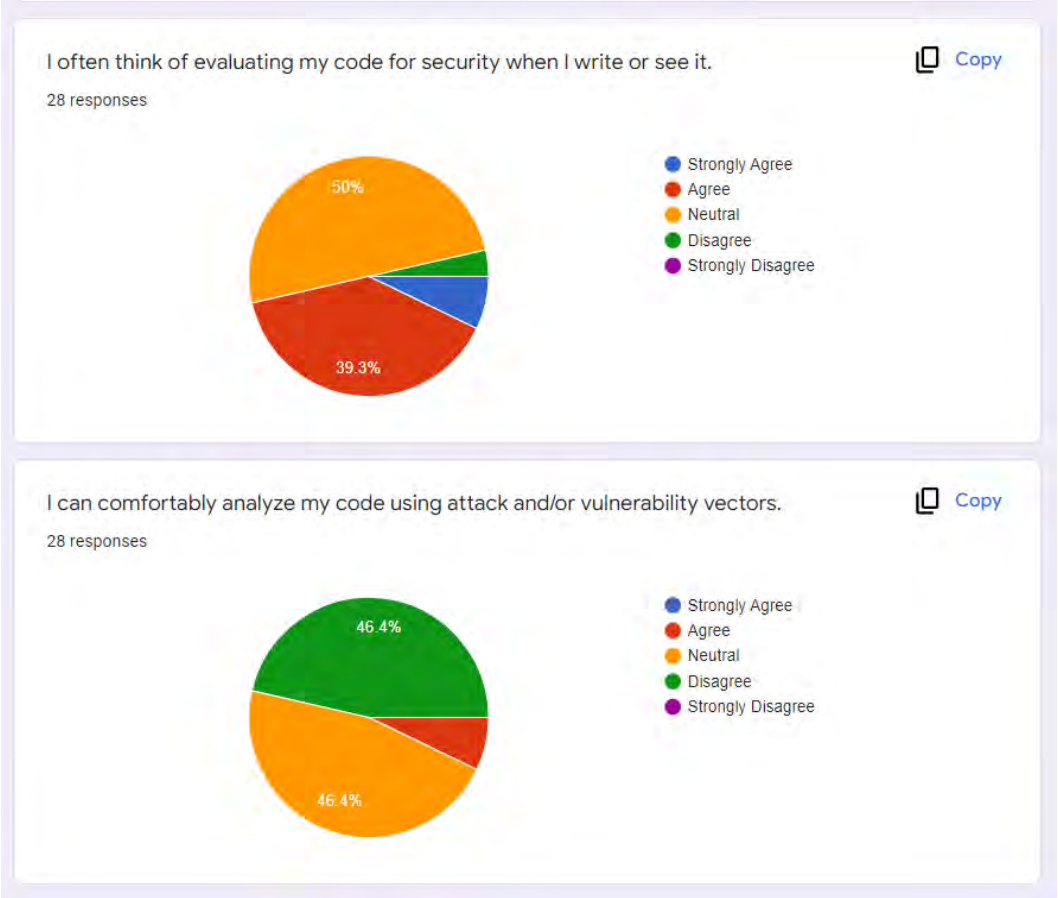
None of our students responded to any of our statements with "Strongly Disagree", indicating that our student population was at least marginally familiar with cybersecurity, including our beginning learners. However, 46.4% of our students indicated "Disagree" to the statement "I can comfortably analyze my code using attack and/or vulnerability vectors", our highest concentration of the "Disagree" response within the survey. This demonstrates an overall unfamiliarity with secure coding as a mindset within our student population.

Further exemplifying this unfamiliarity, our questions that garnered the most "Neutral" responses were our two secure coding mindset questions (see figure 5). For the same statement mentioned in the previous paragraph, 46.4% of our students responded "Neutral", making a total of 92.8% of students who either marked "Neutral" or "Disagree" to their ability to analyze their code with attack/vulnerability vectors. 50% of students also responded "Neutral" to the statement "I often think of evaluating my code for security when I write or see it", indicating that a majority of our student population does not actively prioritize security while coding.

When it came to passive security awareness, however, we received a much more positive response. In response to the statement: "While navigating the Internet, I have a high level of awareness of web attacks and/or web attack vectors", 50% of students indicated "Agree" and 25% indicated "Strongly Agree", making a total of 75% of our population that claimed security awareness during their own browsing. A similar level of enthusiasm was garnered for our question on cybersecurity interest, with 42.9% responding "Agree" and 39.3% responding "Strongly Agree" to our statement "I am currently interested in topics of cybersecurity and secure coding", for a total of 82.2% positive responses.

Our final question regarding access to cybersecurity resources was the most split. The highest concentrated response was tied between "Neutral" and "Agree" at 35.7% each, followed by "Strongly Agree" and "Disagree" which were also evenly split at 14.3% each. This forms an excellent representation of the mixed background of our student participants, some of which are cybersecurity beginners and others

*Figure 5: Image of a pie chart representation of responses to our first two questions, demonstrating the high level of "Neutral" responses*




enthusiasts, as well as a general need for our curriculum to provide more reliable cybersecurity resources to our students across the board.

## Post-Survey Results

We received a total of 11 responses from our student participants on the post-survey of five, Likert-style questions with additional open-ended questions. This accounts for 11 out of our total of 19 active users, three of which had completed every challenge available on the site (as an additional note, many of the top performing students within our group, including two of our participants who completed every challenge, were female). We did not collect data on why some students did not complete every challenge, but our assumptions are that some did not have enough motivation to complete all challenges (seeing as it was not a requirement to do

so), others may have been overloaded with work from classes, and still, others may have only attempted challenges written in programming languages they were already familiar with. The focus of our post-survey was to judge not completion or proficiency with the challenges, but to evaluate changes in our students' mindset regarding secure coding as a result of them.

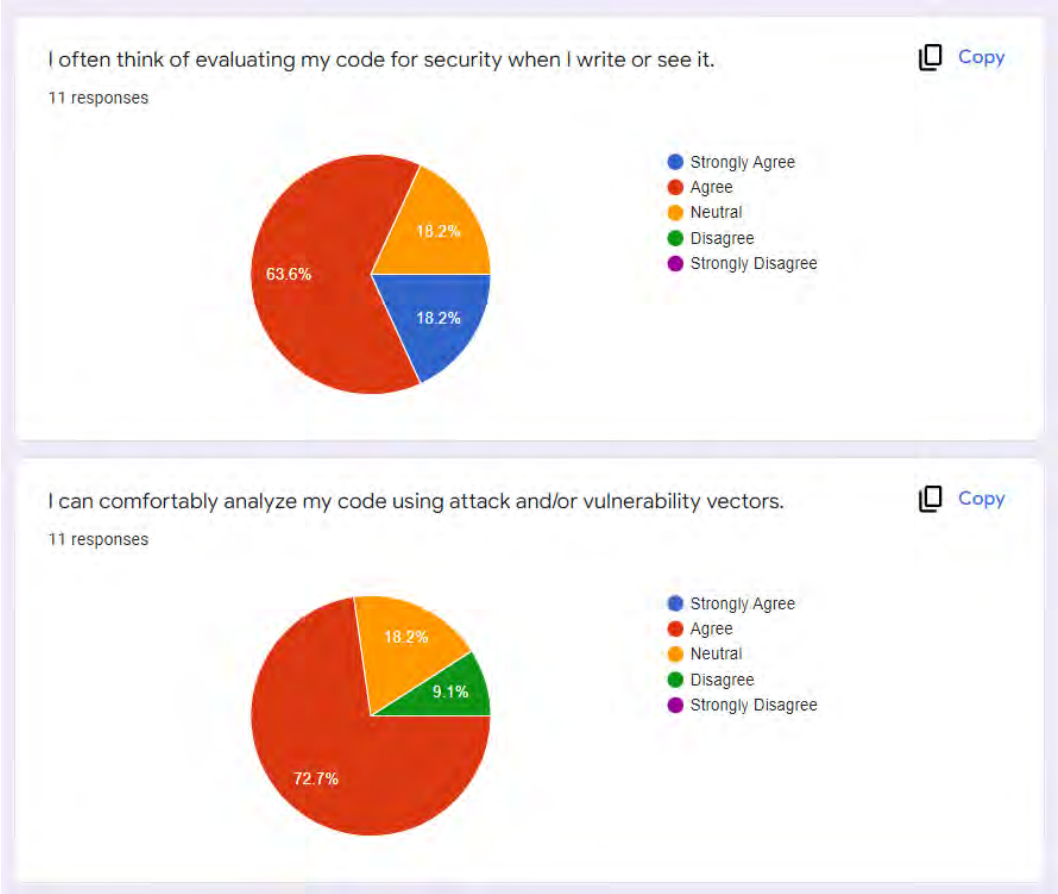
As such, the Likert-scale questions on the post-survey were identical to the pre-survey set, paired with additional open-ended questions that encouraged students to recall their experiences with the CTF challenges and secure coding concepts relevant to each question. A final open-ended question was created at the end of the survey, asking for additional thoughts from our participants.

To begin with a comparative analysis of the Likert-style questions, we can again say that none of our participants responded with "Strongly Disagree" to any of our statements. Further, only one statement garnered any "Disagree" responses at all: "I can comfortably analyze my code using attack and/or vulnerability vectors". The concentration of our "Disagree" responses was marginally lower on this question than in the previous survey, with only 9.1% of participants selecting it. In total, compared to the 92.5% of students who previously responded to this statement with either "Neutral" or "Disagree", only 27.3% of students in the post-survey responded with either of those categories, while 72.7% responded "Agree", showing a marked increase in secure coding familiarity and comfort among our participants (see figure 6).

Our second secure coding mentality statement - "I often think of evaluating my code for security when I write or see it" - saw a similar positive increase, going from a majority response of 48.1% "Neutral" to a majority of 63.6% "Agree", with the rest of the percentage split evenly at 18.2% between responses for "Neutral" and "Strongly Agree". The increase in our level of "Strongly Agree" responses, in particular, going from 7.4% in the pre-survey to the even split of 18.2% in the post-survey, also showcases a notable improvement in our participants' secure coding mentality (see figure 6).

Similar to our results in the pre-survey, our most positive responses came from the statement "While navigating the Internet, I have a high level of awareness of web attacks and/or web attack vectors". However, whereas the pre-survey reported a total of 74% of responses being either "Agree" or "Strongly Agree", the post-survey was 100% comprised of these two responses, with 63.6% of participants choosing "Agree", and the remaining 36.4% choosing "Strongly Agree". This shows not only a maintaining of student confidence in security awareness, but a solid increase in it that has eliminated indications of discomfort and unfamiliarity with the subject.

*Figure 6: Image of a pie chart representation of responses to our first two questions, demonstrating the dramatic increase in positive responses to our secure coding mentality questions.*

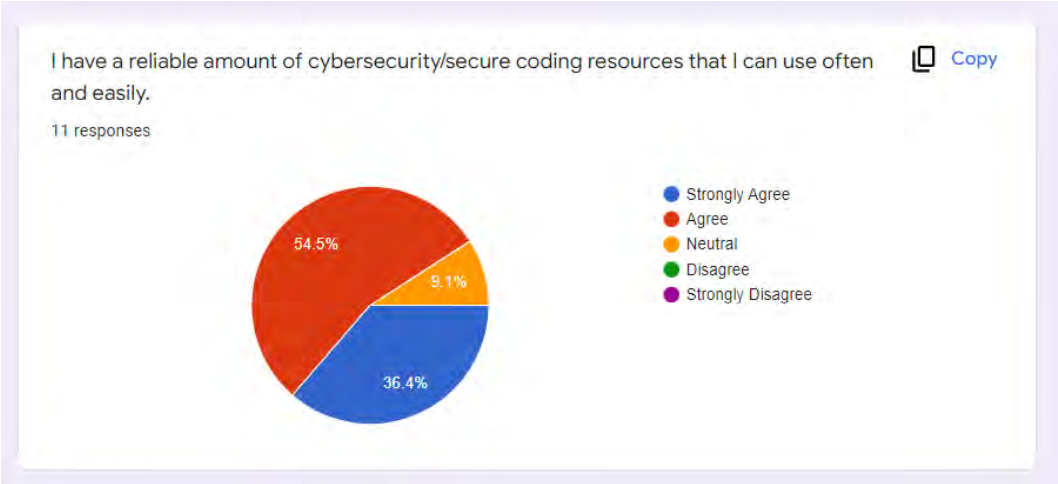



Our statement on cybersecurity interest saw a similar increase in student confidence, but not at a drastic level. Responses here were split at 45.5% between "Agree" and "Strongly Agree", with the remaining 9.1% being "Neutral". We saw an improvement from the pre-survey in how "Strongly Agree" rose to equal the amount of "Agree" responses indicated, creating a total of 91% positive responses compared to the pre-survey's rating of 81.4%. This demonstrates a steady increase in cybersecurity interest from our students after completing the challenges.

Finally, our statement on cybersecurity resources was highly different from the results of the pre-survey. No longer resulting in the most split responses, this statement garnered a solid majority of 54.5% responses of "Agree", with 36.4% of other responses being "Strongly Agree" and only 9.1% of responses being "Neutral". The majority response changed from being "Neutral" to "Agree", and "Disagree"

disappeared entirely from received responses. This shows another drastic increase in our students' confidence, this time representing their increased access to reliable cybersecurity resources (see figure 7).

*Figure 7: Image of a pie chart representation of responses to our final question, indicating the far more even responses received in the post-survey, compared to the split responses of the pre-survey.*




Overall, we saw a greater amount of confidence from our student participants across the board in their secure coding mentalities, passive security awareness, cybersecurity interest, and resource access. A note on these changes, however, is that this increased confidence may have come in part from the concurrently taught network security class these students were in, rather than as a direct result of their participation in the CTF Challenges themselves. When examining the open-ended responses that we received, we gain a much clearer picture of what benefits students perceived that came specifically from the CTF challenges.

As open-ended questions were made optional in the post-survey, not every student submitted a response for each one. Collectively, we received 38 open-ended responses to the 6 open-ended questions posed. Out of these responses, the most impactful to our understanding of the CTF Challenges' effect were the following:

- *"CTF challenges are now an additional learning resource that are definitely getting me out of my comfort zone with NS but really get me thinking."*
- *"Learned more about penetration testing and vulnerabilities in code than if it hadn't been in the course. The high bonus point incentive also really incentivizes people to work hard on it."*

- *"The code analysis was interesting as well, it made me a lot more aware about how data is stored when writing programs"*
- *"Need more challenges"*
- *"The one thing that was a bit difficult...is that the hint system is a bit too technical at times...It probably is expected that we were familiar with certain tools or concepts, but it would be nice to kind of dumb down some of the hints or make their explanations a bit more expansive"*
- *"Maybe it would be chill to have like a tutorial or prompt system to interact with the user upon entry or throughout the challenges to keep them engaged/on task."*
- *"you could have the point locked hints be for how to use [freely provided] tools, where to refer in the article/tutorial linked previously, and then how to acquire the answer (but of course without saying it explicitly)"*

These comments showed us a variety of things about our program. One of the most surprising was that some students considered the challenges a practice resource, and indicated a desire to return to the challenges in the future for more practice. This went hand-in-hand with a request several students mentioned for more challenges, both as general additions to existing content, and as specific additions of certain programming languages/security issues not as well represented in the current challenges. While we had intended our program to be used as a one-semester, concurrent addition to the network security course, it seems our students would appreciate it more as a continuous resource they can access post-course, with increasing content and additional challenges being added for further practice. Whether this would be feasible requires deeper consideration, and would also necessitate a restructuring of the extra-credit incentive given to students, as completing all challenges on a continuously expanding platform is not fair to future students.

From seeing these comments, it is also clear that the extra-credit incentive is a necessary part of this concurrent program. Specifically, the "high" nature of the incentive that allows students to potentially skip their midterms and finals is a massive draw for participants. While many of them were eager to learn more about cybersecurity outside of the scope of the network security course, these incentives likely served much better to garner a wider pool of participants at the beginning.

Our inclusion of source code also appeared to be a good idea based on some of the responses received, encouraging student participants to think more deeply about different aspects of their code "when writing programs". This directly tied into our

objectives for getting students to think with a secure coding mindset while they are writing their own code. However, we did not receive any responses from students who considered themselves beginners about how much more accessible the source code made the challenges for them. It remains unclear as to whether the source code increased or decreased the accessibility of our challenges and the amount of them new students could complete.

On the topic of accessibility, the final takeaway that we learned from the above comments was that our hint system was not as helpful as it could have been, particularly for beginning students. The technical language of the hints made them more confusing, and the minimalist approach we took with adding links to external sources also garnered some frustration. While the extra sources that we offered students were appreciated, beginner students sometimes did not know how to navigate these sources and found them to be a further layer of complexity keeping them from figuring out their next step. One student suggested making the external resources "free", that is, provided initially in each challenge's description rather than only available in point-locked hints. The hints could then be unlocked via points with extra advice on how to utilize these resources, or where to find pertinent information for the challenges at hand. Another comment from above gave suggestions on making the interface of the website itself more user-friendly for beginners, with a potentially dynamic website set-up that can keep students engaged as they begin to learn how to complete the challenges. This student described the initial website as "overwhelming", indicating another weakness with our current set-up for how accessible the CTF challenges are to students.

While the comments discussed in-depth above were the most impactful for us, it is also worth noting that throughout the open-ended responses, students often made direct references to practices they had learned during the CTF challenges or as a result of the challenges. For example, "After CTF-> [evaluating code for security means] any type of overflow that would cause a leak, double checking user input values, having up to date patches if using certain software". One student even directly referenced a piece of code from one challenge as an example of a vulnerability: "flag = "bananas";". This demonstrated clearly for us that many of these students were considering the CTF challenges in their thought processes for cybersecurity and secure coding, and that it had framed examples of security vulnerabilities for them to be aware of.

Based on the wording of responses like these as well as the impactful comments above, we have seen that the CTF challenges proved to be a useful resource for framing and improving our students' security mindsets.

# CONCLUSIONS AND FUTURE DIRECTIONS

Ultimately, based on the results enumerated above, we conclude that our CTF challenges were moderately effective at improving the secure coding mentality of our students. Students were on average more likely to consider the security of their code while writing it and to consider security vulnerabilities while examining other code or browsing through websites. Most of our students believed that the challenges were a fun and engaging way to let them practice secure coding concepts in a hands-on manner, supplementing the content of the network security class they were enrolled in. Many of our students demonstrated increased confidence in their ability to recognize, patch, and/or exploit security vulnerabilities, as well as increased awareness of cybersecurity resources they could use to practice such activities post-challenge.

Based on the experience of developing CTF secure coding challenges, we would like to encourage others to take a look at real-world vulnerabilities (for example, on the Vulmon search engine that we used (Vulmon, 2022)), learn about the impact and the main exploitation vector, and develop a challenge in any modern language that would mimic the real vulnerability in a simplified form.

Moving forward, one of the obvious weaknesses of our study is our limited sample size for students who attempted our CTF challenges. Additionally, it should be noted that there is a possibility of non-response bias as the pre-survey and post-survey were anonymous and not required (hence, there is a difference in the number of responders in the pre-survey and post-survey). While we wish we could have increased that size, our class sizes are extremely small at our liberal arts institution, and there are not many electives offered frequently where these challenges could have been introduced in parallel in the same semester. We would recommend those who take our work further test challenges like these on a greater number and variety of students, for an extended period of time. Also, it is likely that requiring the students to complete the post-survey would facilitate better analysis and avoid the challenges of non-response bias.

Regarding the CTF challenges themselves, further improvements can be made which include expansive content, a platform navigation tutorial, and more beginner-friendly hints and resources. In general, our current system would ideally be made substantially more user-friendly for beginner students and would incorporate more pedagogically-helpful tools for our students to encourage them to learn the challenge material in smaller, scaffolded steps. Particularly, we would want to create this scaffolding within our hint system, while leaving the challenges themselves similar to traditional CTF challenges, without any additional help for the students attempting to solve them. We will also consider making the CTF challenges an

ongoing platform that will be consistently expanded for students who intend to use it as a continuous training/practice resource. In the future, we would expand the platform by incorporating more challenges in other coding languages, increasing the difficulty of existing challenges and relaunching them as new levels, or introducing new concepts from cybersecurity not as greatly emphasized in our current challenges, such as database attacks.

# REFERENCES


Bellovin, S. M., & Bush, R. (2002). Security through obscurity considered dangerous.

Beunardeau, M., Connolly, A., Geraud, R., & Naccache, D. (2016). White-box cryptography: Security in an insecure environment. *IEEE Security & Privacy*, *14*(5), 88–92.

Chi, H., Jones, E. L., & Brown, J. (2013). Teaching secure coding practices to stem students. *Proceedings of the 2013 on InfoSecCD'13: Information Security Curriculum Development Conference*, 42–48.

Chothia, T., & Novakovic, C. (2015). An offline capture the flag-style virtual machine and an assessment of its value for cybersecurity education. *2015 {USENIX} Summit on Gaming, Games, and Gamification in Security Education (3GSE 15)*.

Chung, K. (2017). Capture the flag platform [https://ctfd.io].

Cobb, S. (2016). Mind this gap: Criminal hacking and the global cybersecurity skills shortage, a critical analysis. *Virus Bulletin Conference*, 1–8.

Davis, A., Leek, T., Zhivich, M., Gwinnup, K., & Leonard, W. (2014). The fun and future of {ctf}. *2014 {USENIX} Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*.

Edwards, C. (2014). Researchers probe security through obscurity.

Ford, V. (2022a). Deployed free secure coding ctf platform [https://ctf.vford.com].

Ford, V. (2022b). Secure coding ctf platform source code [https://github.com/vitalyford/secure-coding-ctf].

Gasiba, T., Lechner, U., Pinto-Albuquerque, M., & Zouitni, A. (2020). Design of secure coding challenges for cybersecurity education in the industry. *International Conference on the Quality of Information and Communications Technology*, 223–237.

Gonzalez, H., Llamas, R., & Ordaz, F. (2017). Cybersecurity teaching through gamification: Aligning training resources to our syllabus. *Res. Comput. Sci.*, *146*, 35–43.

Gonzalez, H., Llamas, R., & Rivas, O. M. (2019). Using a ctf tournament for reinforcing learned skills in cybersecurity course. *Res. Comput. Sci.*, *148*(5), 133–141.

Mirkovic, J., & Peterson, P. A. (2014). Class capture-the-flag exercises. *2014 {USENIX} Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*.

Paulsen, C., McDuffie, E., Newhouse, W., & Toth, P. (2012). Nice: Creating a cybersecurity workforce and aware public. *IEEE Security & Privacy*, *10*(3), 76–79.

Saxena, A., Wyseur, B., & Preneel, B. (2009). Towards security notions for white-box cryptography. *International Conference on Information Security*, 49–58.

Schreuders, Z. C., & Ardern, L. (2015). Generating randomised virtualised scenarios for ethical hacking and computer security education: Secgen implementation and deployment.

Taylor, B., Bishop, M., Hawthorne, E., & Nance, K. (2013). Teaching secure coding: The myths and the realities. *Proceeding of the 44th ACM technical symposium on Computer science education*, 281–282.

Taylor, B., & Kaza, S. (2011). Security injections: Modules to help students remember, understand, and apply secure coding techniques. *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, 3–7.

Vulmon. (2008). Cve-2008-4160 wordpress sql truncation vulnerability [https://vulmon.com/vulnerabilitydetails?qid=CVE-2008-4106&scoretype=cvssv3].

Vulmon. (2014). Cve-2014-0160 heartbleed vulnerability [https://vulmon.com/vulnerabilitydetails?qid=CVE-2014-0160&scoretype=cvssv3].

Vulmon. (2022). Vulnerability search engine, from products to vulnerability types [https://vulmon.com/].

# APPENDIX

*Figure 8: Image depicting the number of times each of our challenges were solved by different participant teams.*

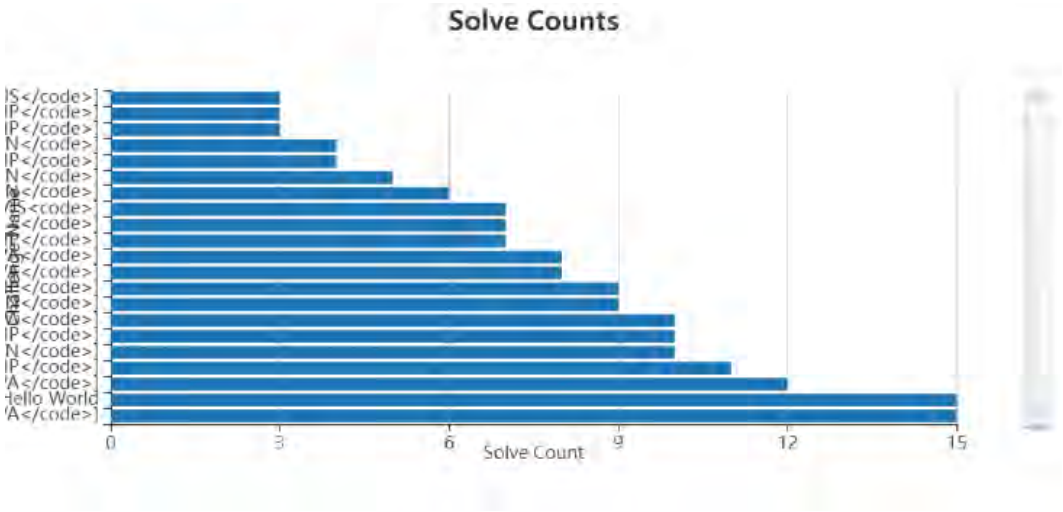



*Figure 9: Image depicting the concentration of score ranges across our participant teams, showing great variation in the ranges achieved.*

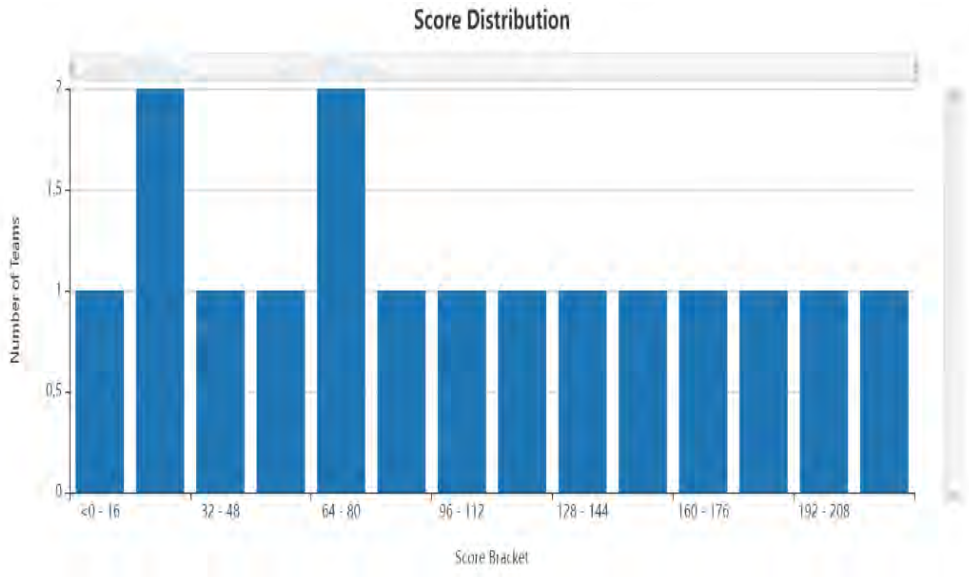


*Figure 10: Image depicting the percentage of participant teams who were able to solve each challenge.*

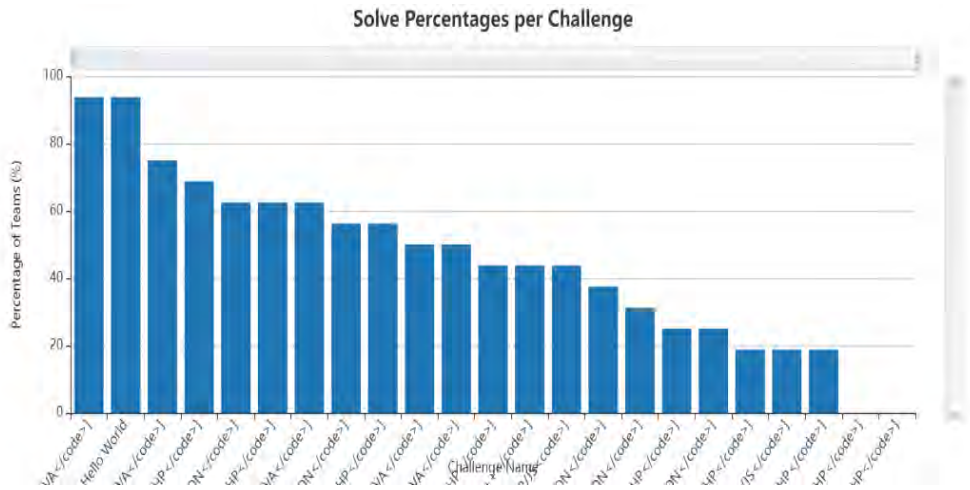



*Figure 11: Image depicting the percentages of solution submissions we received from participant teams that were either solves or fails.*

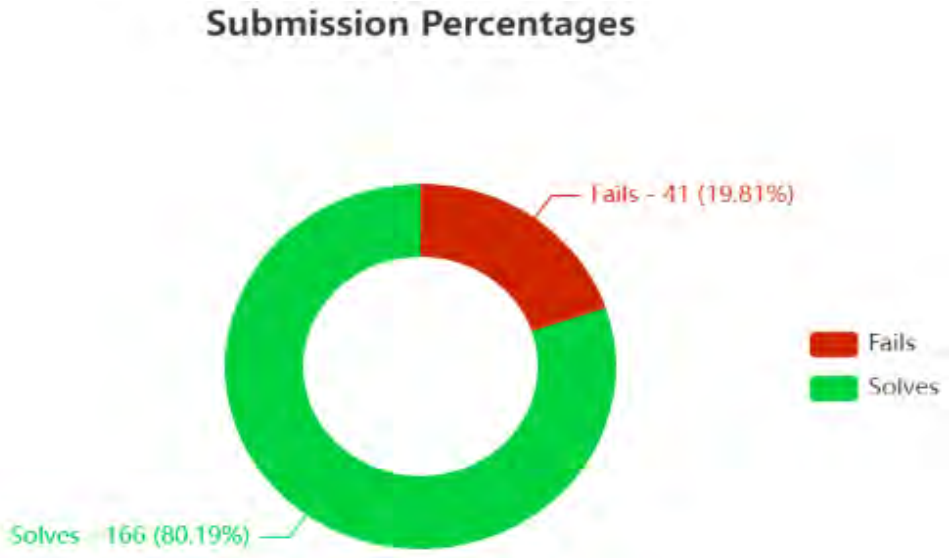


*Figure 12: Image depicting the percentage of challenges that were from different weeks in the CTF platform (the later weeks indicate higher difficulty).*