Analysis and Exercises for Engaging Beginners in Online CTF Competitions for Security Education

Tanner J. Burns[1], *Samuel C. Rios*[1], *Thomas K. Jordan*[1], *Qijun Gu*[1], *Trevor Underwood*[2]
[1]*Department of Computer Science, Texas State University, San Marcos, TX 78666*
Email: {tjb102,scr3,tkj15,qijun}@txstate.edu
[2]*Netspend Corporation, Austin, TX 78768*
Email: tunderwood@netspend.com

Abstract

Cybersecurity competitions are getting more attention as a prominent approach of computer security education in the past years. It is vital to look into better ways to engage beginners in the competitions to improve computer security education. This work collected and analyzed the solutions of about 3600 Capture The Flag (CTF) challenges from 160 security competitions in the past three years. This work identified the security issues that are the most concerning to industry and academia and enumerated the security tools and techniques that are used the most by players. Based on the analysis, this work presented a set of computer security exercises as a downloadable tool package for beginners to try out in an introductory computer security course.

1 Introduction

Cybersecurity competitions in recent years have attracted many students and professionals interested or working in the areas of computer security. Throughout this research, we collected the data of past competitions from 2011 to 2016 from the archives of CTFtime.org [1] to study the main characteristics of the competitions. These competitions were used for education, training, and recruitment. In universities, some of the competitions were incorporated into the computer security education curriculum to enhance the interest in a cybersecurity career among the students with hands-on gamificated exercises and to produce the next generation of cybersecurity professionals.

A great deal of effort has gone into making the competitions better. However, it is still vital to look into better ways to engage beginners in the competitions [12,15,18]. Beginners are often dissuaded from the competitions due to the frustrating experiences of their first attempts. Meanwhile, it is in the student's best interest to continuously participate in the competitions so that they can accumulate security knowledge and skills. To address these issues and help beginners, this study intends to identify the essential skills and provide suitable training resources for beginners to study and practice by themselves.

As shown in Figure 1, online competitions (more than 60%) have been the dominant way to host the competitions. Among the online competitions, about 94% are of jeopardy style, namely Capture The Flag (CTF), where players use offensive techniques to solve security challenges. Compared with other forms of competitions, online CTF competitions are more accessible for beginners. Accordingly, this work provides a training platform and a set of exercises that mimic the online CTF competitions for beginners to gain initial similar experiences.

The main question this work tries to answer is what security knowledge and skills should be included in the exercises for beginners. Rather than designing the exercises based on our past security teaching experiences, we chose to study the past CTF competitions to identify the security issues that are the most concerning to industry and academia and enumerate the security tools and techniques that are used the most by players. Then, based on the analysis, we design our exercises. This data-based and analysis-based approach will better reflect the truly needed security skills for beginners to grow their interests and skills in computer security.

To understand the kinds of skills and techniques to solve these security challenges, we collected and analyzed the solutions of about 3600 security challenges from 160 security competitions in the past three years. Many players voluntarily posted their solutions as "writeups" that provide step-by-step solutions and their thought processes. The writeups are excellent sources for beginners to study and follow. Beginners can find the writeups on Github [2], CTFtime [1], players' personal blogs or websites, and so on. After comparing these sources, we chose to collect writeups from Github [2], where more writeups were posted and better organized.

The contribution of our work has two folds. First, we showed the main characteristics of the past security chal-

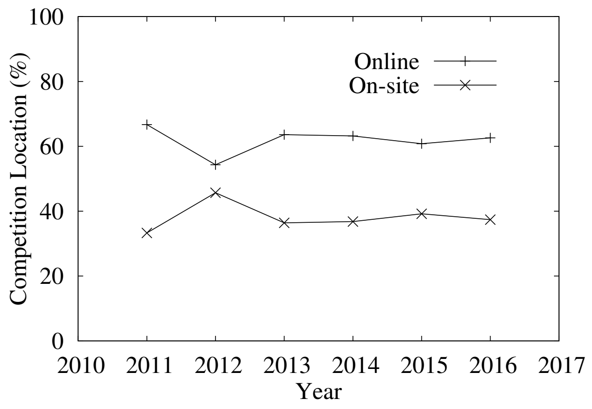

Figure 1: Competition Locations

lenges, summarized the main security issues concerned in the security community, and highlighted the main knowledge and skills used in the competitions. Second, we provided a training platform based on PicoCTF [6] and deployed a set of exercises based on the analysis of the past security challenges. The platform and exercises are centered around beginners and enable beginners to study and practice by themselves.

The rest of the paper has the following sections. Section 2 summarizes the related works in the literature. Section 3 discusses the choice and setup of the CTF platform. Section 4 presents the analysis of the past CTF challenges and the exercises included in our platform. Section 5 shows the results of using the platform and the exercises in an introductory computer security course. The paper concludes in Section 6.

2 Related Works

In recent years, competitive security games, especially Capture-The-Flag games, have emerged as a new educational approach that gamificates computer security education to attract and engage more students into computer security areas [9, 11, 13, 14, 16, 17]. Many CTF games are online and easier for students to participate, practice and learn on specific security subjects. To help beginners to participate, educators and researchers have invested a great amount of effort on developing CTF tools and redesigning CTF-based curricula to make the CTF games more accessible and useful to students.

Online CTF competitions are hosted on a variety of platforms. Several hosts of the competitions have published their CTF platforms [3–7]. We adopted PicoCTF [6, 10] as our platform to deploy the CTF exercises. The platform has well designed features including user management, web interface, problems setup, problems grading, statistics of players and teams, and so on. It is also well documented and easy for us to deploy exercises. It later was improved [8] to include automatic flag generation to distinguish participating teams. Similar to the approach in [11], we revised the PicoCTF platform so that students can download the platform as an offline virtual machine to practice by themselves.

Educators also tried to make CTFs more suitable for beginners. In [12], six factors were analyzed to find the reasons that security competitions were very hard for beginners. It was found that most competition challenges were not designed for beginners in the first place, and thus led to beginners quickly becoming stuck and giving up. In [15], educators designed a set of small-scoped and hands-on exercises and in-class discussions to gently introduce beginners to security competitions, rather than simply exposing them to hard problems that they cannot solve. In [18], educators divided security problems into several levels accompanied with a few hints at each level as well as a recommended solution. Beginners could opt to take the hints and the solutions. We notice that these works did not pay sufficient attention to what exercises are suitable and important to beginners. We did a comprehensive analysis on the past CTF challenges to help beginners understand the characteristics of the CTF challenges and the prominent skills and areas they need to learn in order to participate in the competitions.

3 Setup of Platform and Exercises

3.1 Platform Choices

We provided all exercises inside a standalone Virtual Box (VBox) platform so that beginners can download the platform to practice on their own computers. Many beginners do not have experiences with online CTF competitions at all. They do not know how to read questions and hints of security challenges, how to manually or programmably interact with the services of the challenges, how to submit flags, how to check their progresses, and how to form teams online. Although these obstacles appear trivial to experienced players, we found it is necessary to provide real competition experiences to beginners so that they can better focus on security challenges.

We built the platform based on the PicoCTF platform [6]. For beginners to practice on their own computers, we package the CTF management and the exercises inside one CTF platform. There are a few alternative open source CTF platforms, such as OpenCTF [5], CTFd [3], FbCTF [4], TinyCTF [7] and so on. From the perspectives of beginners, all of the platforms we studied have similar features, for example, user management, web interface, problems setup, problems grading, statistics of players and teams, and so on. Although beginners will have similar experiences practicing on these platforms, the process of deployment (i.e. installing these platforms and setting up exercises) varies greatly.

Table 1: Comparison of CTF Platforms

	Installation	Language	Documentation
PicoCTF	Vagrant ✓	Python ✓	Good ✓
OpenCTF	Docker ✓	Python ✓	Simple
CTFd	Native	Python ✓	Simple
FbCTF	Vagrant ✓	PHP	Good ✓
TinyCTF	Native	Python ✓	Simple

To choose a proper platform, we compared three factors regarding the deployment and development of these platforms as shown in Table 1. The first factor is how a platform is installed and setup. The native approach is to directly install a CTF platform as a software in a computer. It usually requires extensive technical skills to install and configure, and thus is challenging to beginners. It is also risky because the host machine runs a CTF platform with exploitable services. Vagrant and Docker are the two latest technologies that are used to install a CTF platform inside a virtual machine or a software container. They add a protection layer that separates the CTF platform from the host machine. They are also easy for beginners to install by themselves. The second factor is what programming language is used by the CTF web application. A CTF platform runs a web application for participants to interact with the system. The web application manages users, security challenges, grading, statistics and so on. Among the platforms we studied, Python and PHP were the two web programming languages utilized. Python-based web applications use the Flask framework, which is a lightweight Python web framework and easy for us to manage, modify and extend our own features into the CTF platform. The third factor is how well a platform is documented. We looked for two sets of documentations. One is the installation instruction, which is easily accessible for all of the platforms. The other is the guide for adding new security challenges and customizing or tweaking features. Not all platforms documented this aspect well. Some platforms simply ask developers to follow the example challenges to figure out how to add their own challenges.

After considering these three factors, we chose PicoCTF as the base platform. But, we did not want beginners to run Vagrant to setup their own PicoCTF. We built a VBox machine based on PicoCTF that beginners can download to run in their own computers. So, beginners do not need to tackle occasional technical issues of using Vagrant and can quickly run the VBox to practice.

3.2 Exercise Categories

Most CTF competitions divided their security challenges into different categories. Similar challenges were grouped in the same category. Players often formed teams that could quickly divide the workload among team members based on the problem categories in competitions. Most competitions have 5 to 7 categories and most categories have 4 to 5 challenges per category. We wanted beginners to initially focus on and obtain skills from a selected set of categories. We analyzed the 909 category names used in the 160 competitions and identified 77 unique category names in these competitions. We concluded it was best to provide exercises in six categories: "crypto", "web", "reverse", "forensic", "pwn" and "misc"

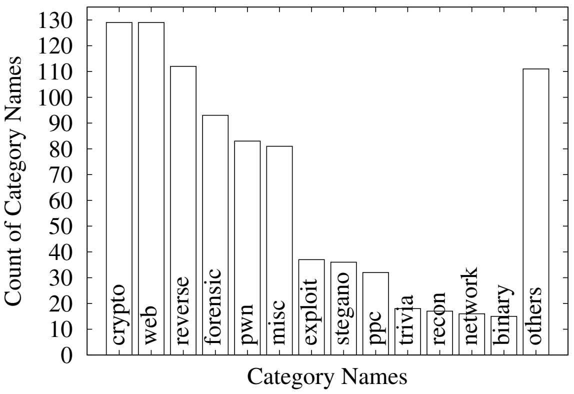

Figure 2: Histogram of Category Names

Figure 2 shows the histogram of the category names. We can see that the category names can be divided into three groups according to their counts. The first group includes six category names that we use for our exercises. They represent the major categories of security challenges in CTF competitions. The second group includes "exploit", "stegano", "ppc", "trivia", "recon", "network" and "binary". The category names of the second group were often used as substitutions of the first group, or were used to make finer categories on some specific topics. Often the challenges of the second category group overlap with the first category group. For example, the "exploit" challenges often belong to either "pwn" or "web" in the first group, and the "binary" challenges often belong to either "reverse" or "pwn". The remaining sixty-four category names form the third group. They only appeared in very few competitions, but can be classified to the categories of the first or second groups.

3.3 Exercise Difficulty Levels

A CTF competition is usually made of security challenges at different difficulty levels. Although the difficulty levels of different CTF competitions vary greatly, we defined three levels for beginners as seen below. We studied the solutions of the past security challenges and reclassified them accordingly. Table 2 shows the numbers of the security challenges of each category at each difficulty level of the past CTF competitions.

1. *Easy*: A challenge can be solved with one or two tricks and tools. A beginner can often solve the challenge by themselves right after reading the writeups.

2. *Medium*: A challenge can be solved with three or more tricks and tools. After reading the writeups, a beginner can solve the challenge with extra efforts, such as reading additional documents.

3. *Hard*: A challenge can be solved with in-depth tricks and sophisticated tools. A beginner can hardly understand the writeups and cannot solve the challenge even after reading the writeups.

We can see that the "misc" category has significant fewer challenges than the other categories in security competitions, because the "misc" challenges are often used to test the general programming, math and problem-solving abilities of the players. For the five security categories, "crypto", "web" and "forensic" appear to be easier than "reverse" and "pwn". The hard challenges are less than a quarter of the total problems in the "crypto", "web" and "forensic" categories, while they are more than 40% in the "reverse" and "pwn" categories. Most challenges of the "reverse" and "pwn" categories require the skills to disassemble a binary executable, analyze the components and workflows inside the executable, and then reconstruct a higher level abstraction of the executable. The "pwn" category is even harder than "reverse" in that it requires the skills to identity a flaw and exploit the flaw in the remote service after reverse engineering binary executable. These skills usually require beginners to take more training to master.

In total, three quarters of the security challenges were easy and medium ones. Our exercises were chosen from the easy and the medium security challenges. We did not include hard challenges in our exercises. We think, by practicing on these selected exercises, beginners can develop the essential skills to solve a majority of easy and medium challenges in competitions to gain confidence and successful experiences. Once they have established their own expertise, they can develop in-depth skills for those hard challenges in future.

4 Exercises

We have collected a large set of the past security challenges. Based on our analysis of these challenges, we selected a limited set of challenges that represent the most concerned security issues in the competitions. We think these challenges also reflect the most concerned security issues in industry and academia. We created the exercises based on the selected challenges. They help beginners build their security knowledge and skills. In this section, we describe the analysis of these challenges and the chosen exercises in the six categories.

Table 2: Difficulty Levels

Category	Easy	Medium	Hard
crypto	192 (48%)	129 (32%)	83 (21%)
web	152 (41%)	150 (40%)	70 (19%)
reverse	77 (22%)	131 (38%)	136 (40%)
forensic	263 (50%)	186 (35%)	79 (15%)
pwn	66 (19%)	138 (39%)	148 (42%)
misc	96 (48%)	67 (34%)	35 (18%)
total	846 (38%)	801 (36%)	551 (25%)

4.1 Coding

Coding is one of the fundamental skills required to solve most security challenges. We provided a few exercises in the "misc" categories, dedicated for training beginners on necessary coding skills. Many security challenges provide a few programs in source code or binary. Players need to figure out clues and flaws in the programs to solve the challenges. In addition, players need to make programs to process data and interact with flawed services. All these activities need players to be proficient in coding. In particular, we identified several must-have coding skills for beginners that are focused on security-related data processing operations. We included these skills in our coding exercises to improve the beginners' proficiency and efficiency on data processing.

The first coding skill is number and string conversion. Most security-related data (text and numbers) is represented as ASCII characters, hexadecimal numbers, Base64 strings and so on. We notice that beginners often spent unreasonable long time to use and process these data. It is very necessary for beginners to master a few approaches that can quickly perform hexadecimal and binary conversions, string and number conversions, large number arithmetic, Base64 encoding and decoding, string splitting and concatenation, and so on. Therefore, beginners can have more time to focus on the challenges. The second coding skill is file manipulations. Beginners often need to open files to read and analyze data in a programmable and automated way. Combined with the first coding skill, beginners should be able to quickly write a program or a script to process data in files. The third coding skill is network programming. In many challenges, beginners need to connect to a service, and receive and send data to the service. Doing this with the normal Netcat or Telnet program manually is slow. The services of the challenges are usually set to time out quickly too. Hence, beginners need to learn networking programming to create services, make and send arbitrary packets to remote servers, and process packets received from remote servers.

Table 3: Groups of Cryptographic Algorithms

Groups	Problem Ratios	Cipher Counts	Top 2 Ciphers
Custom	37.3%		XOR
Symmetric	34.4%	36	AES, Caesar
Asymmetric	21.3%	10	RSA, ECC
Hash	5.3%	5	MD5, SHA1/2
Misc	1.7%	4	DSA, SSL

Table 4: Top Executable Program Types

Rank	Reverse		Pwn	
1	X64	33.9%	X64	31.7%
2	X86	28.7%	X86	31.3%
3	Java	9.1%	C	16.7%
4	PE32	6.3%	Python	8.2%
5	Python	4.5%	Bash	3.2%
6	Others	17.5%	Others	8.9%

We incorporated these three skills into our exercises where beginners have to develop these skills to solve the challenges. We also suggest beginners to program in Python, which is the dominant programming language in the solutions of many competition challenges, and is easy for beginners to learn. Beginners can quickly make some Python scripts and run them to test solutions. There are a lot of supporting libraries to handle networking, web, strings, numbers, arithmetics, penetration, and various files in Python.

4.2 Cryptography

The "crypto" category includes many challenges based on a variety of cryptographic algorithms. Table 3 summarizes the cryptographic algorithms in five groups. Among custom algorithms, XOR is the most commonly used operation. Symmetric cryptographic algorithms appeared in about a third of challenges. We identified 36 publicly known symmetric algorithms, much more than the counts of the publicly known algorithms in the other groups. Asymmetric cryptographic algorithms appeared in about one fifth of challenges, but has only 10 algorithms. RSA tops in the competitions among all publicly known cryptographic algorithms. Hash algorithms appeared in about 5.3% of the cryptographic challenges. MD5 has well known collision issues and thus appeared in more challenges than SHA1/2. The misc group includes the challenges that were designed based on cryptographic tools, libraries and protocols.

Further, we examined the flaws in the "crypto" category. We found that more than half of the "crypt" challenges have custom cryptographic flaws where designers of the challenges made their own flawed cryptographic algorithms. Compared with the "pwn" and the "web" categories, there are much fewer known flaws in the "crypto" category. For asymmetric encryption, we identified eight RSA flaws, such as weak public keys, Coppersmith's attack, Wiener's attack and so on. For symmetric encryption, we found AES CBC was often exploited in the competitions. For hash, we identified four hash flaws: length extension attack, MD5/SHA reverse lookup, MD5 collision, and PBKDF2 HMAC collision. Other than these main cryptographic flaws, there were a few flaws rarely used in the competitions. For example, RC4's flaw only appeared once in the writeups we examined.

To help beginners, we deployed a few cryptographic challenges at the easy and medium levels that include the mostly used cryptographic algorithms and flaws according to our analysis. With these exercises, beginners will study symmetric ciphers (Caesar, Vigenere and AES), assymetric cipher RSA, and hash algorithms (MD5 and SHA1). They will develop programming skills for encryption and decryption as well as analytic skills for cryptographic analysis, substitution, factorization, hash collision and so on. To solve the challenges,they will need to develop their own programs to exploit the flaws in the cryptographic algorithms.

4.3 Reverse

In "reverse" challenges, flags are usually obfuscated and embedded in executable programs. Players need to read the programs and understand how the programs are executed. Once completed, players find a way to retrieve the flags from the binaries. Static analysis and dynamic analysis are mostly used to solve the "reverse" challenges.

Static analysis asks players to disassemble the binaries to understand the programs with disassemblers (such as Objdump and IDA Pro). Table 4 shows the top types of executable programs. About two thirds of the "reverse" challenges were to reverse engineer executable binaries running in Unix and Linux computers. Hence, beginners need to learn X64 and X86 assembly languages. Beginners can also use decompilers (such as Hex-Rays Decompiler and Hopper) that decompile binaries to C source code, which is much easier to understand than X64 and X86 assembly languages. Java programs are the next most common binary type, because Java programs and Android applications are very popular in web and mobile applications. Many online and offline tools can decompile Java programs and Android applications to Java source code. A beginner then needs to learn Java language to understand the programs. Although there are many other binary types, we think beginners should mainly focus on the Linux executable binaries and Java programs to establish their initial reverse engineering skills. Hence, we deployed a few "reverse" challenges with Linux executable binaries and Android applications

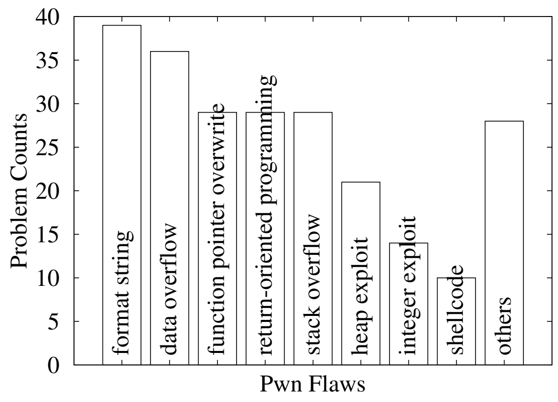

Figure 3: Pwn Flaws

Table 5: Top Data Formats in Data Type Groups

Group	Format Counts	Top 2 Formats
image	8	png, jpg
network	5	pcap, tcpdump
audio	5	wav, mp3
disk	6	dd, ext4
archive	7	zip, tar
dump	6	memory, vbox
text	6	text, c, html
pdf	1	pdf
binary	7	x86-64, x86
video	4	mp4, mpeg
others	16	doc, log

for beginners to practice their static analysis skills.

Dynamic analysis, on the other hand, asks players to execute the binaries, trace the execution, and alter the execution. We observed two dynamic analysis techniques in many "reverse" challenges. One is library trace and system call trace, where beginner need to learn how to use "ltrace" and "strace" to trace and analyze the functions being called during the execution of a binary. Once the key functions are identified, the functions can be overloaded with other custom functions to alter the execution. The other technique is debugging, i.e. running a binary in a debugger. Beginners need to learn stepping and breaking to control the execution, and learn watching and changing variables, memory and registers to alter the execution. Hence, we also deployed a few such challenges for beginners to practice dynamic analysis techniques.

4.4 Pwn

Many "pwn" challenges provide the executable programs that run the flawed remote services to be exploited. Players first need to reverse engineer the programs to discover flaws and then exploit the flaws to hack into the remote services and retrieve the flags. We can see from Table 4 that binaries that can be disassemble to X64 and X86 assembly languages are still the top program types in these challenges. When binaries are not provided, a portion of C source code files that was used to build the remote services are often provided to players. C and Python are the top two coding language to build the flawed remote services.

We found that, besides custom flaws, many "pwn" challenges were built on some well known security flaws, such as the software flaws and mis-configurations published in the Common Vulnerabilities and Exposures (CVE) database. Thereby, we examined what flaws were often used in the "pwn" challenges. Figure 3 shows the types of flaws in the "pwn" category. The counts of the top six flaws are not far apart. Format string is the number one flaw. Following that, we identified four different types of overflow: data overflow, stack overflow, heap overflow (one type of heap exploit) and integer overflow/underflow. Heap exploit also includes a few heap free flaws. If we added up the challenges of all overflow flaws, they would be more than the challenges of the format string flaw. Exploiting function pointers ranks 3rd. For example, overwriting GOT pointers often enables players to run unintended functions. Another exploitation to run unintended code is return-oriented programming (ROP), where a chain of ROP gadgets was constructed to alter the execution of a flawed program. Lastly, shellcode was often exploited in the "pwn" challenges, where players could inject shellcode or exploit existing shell programs to open a remote shell in the target servers. The other flaws were often very specific to some particular programs.

Accordingly, we deployed a few "pwn" exercises based on the top three flaws: format string, data overflow and function pointer overwriting. Beginners can practice to obtain knowledge on these basic flaws and then be able to learn more advanced exploit techniques later.

4.5 Forensic

The "forensic" challenges target extracting information from various types of data files. We identified 69 unique data formats used in the challenges and grouped them to 16 data type groups. Table 5 shows the top 10 data type groups, including the counts of data formats and the top 2 data formats of each data type group. The top 10 data type groups cover more than 94% of forensic challenges.

Accordingly, we deployed a few forensic challenges of the top three data type groups. The first is image challenges, where PNG and JPG pictures are presented with hidden flags. Beginners will learn typical techniques of embedding data in the meta information of images, concatenating multiple image files, tweaking image pixels and son on. The second is network trace challenges,

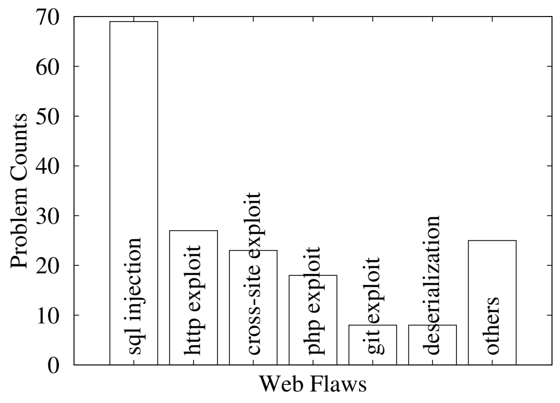

Figure 4: Web Flaws

where PCAP files are provided with hidden flags. Beginners need to follow the network traffic in the traces to find the flags. The third is multimedia challenges, where audio and video files are presented with hidden flags. Multiple data types are used in a few medium-level challenges so that beginners can learn to identify data types through some analysis.

4.6 Web

The "web" challenges require knowledge of web technologies on both server side and client side. On the server side, beginners should familiarize themselves with PHP language, SQL language, and MySQL database, since many websites were setup based on the very popular web framework LAMP (Linux, Apache, Mysql and PHP). On the client side, beginners should be proficient with Javascript and HTML to understand how web pages are dynamically generated and rendered on the client side.

We examined the major flaws that were exploited in the "web" challenges too. Figure 4 shows the types of flaws in the "web" category. Obviously, SQL injection is most commonly exploited. Web sites often need back-end databases to provide data. Meanwhile, web sites often generate dynamic web pages using the data. Hence, the web pages become an attack surface through which malicious SQL statements can be injected into the back-end databases to cause unintended consequences. HTTP exploit ranks 2nd, where the flaws were often due to misconfiguration in web sites, such as local file inclusion, web robot file, relative path and so on. The third is cross-site exploit, where players could inject cross-site scripts and forged requests to exploit the flawed web sites. Among web programming languages, PHP unfortunately became the main target in the competitions due to quite a few well known PHP flaws in the past. Web applications and services developed in Python, Perl and Ruby were far less attacked than those in PHP. GIT was also exploited because many web developers used GIT for version control but left GIT in the web server carelessly. Serialization and deserialization is a process to pass objects in web. Some implementations of deserialization were flawed and thus exploited in the competitions too. All the other web flaws were not common but could appear in web sites if the web sites were not developed or configured properly.

Based on the writeups, we deployed a few "web" exercises that help beginners develop two necessary skill sets. One is to proficiently use CURL and web development tools built in most web browsers to inspect web pages and web traffics. Beginners need to inspect and manipulate cookies, sessions, URLs, form data, JSON data and web agents on the client side. The other is to exploit the top three flaws: SQL injection, http exploit, and cross-site script exploit.

5 Lessons Learned

5.1 Class Settings

To examine the usefulness of the CTF exercises in computer security education, we included the CTF exercises in our undergraduate introductory computer security class. The class has 46 students who were all beginners. They studied security concepts, theories and techniques in class. We assigned the CTF exercises as a part of individual homework assignments, while the other part of homework assignments are traditional textbook questions. Half of the CTF exercises are easy ones, and the other half are medium ones. At the end of the class, we conducted an anonymous survey to get feedback from the students.

We provided partial solutions on the key steps for solving the assigned CTF exercises. The solutions were not complete, and did not disclose the flags. The goals of the solutions were to guide the students with example techniques and to help them overcome the most challenging steps in the exercises. The students needed to figure out the missing steps and completed the exercises by themselves.

Different from some earlier studies [15, 18] where beginners had great interactions with the instructors, we minimized our intervention in this study. The main goals of our CTF platform and exercises are to enable beginners to learn and practice by themselves so that they can obtain technical and psychological confidence by themselves.

5.2 Observations

Installation Issues: Some beginner students encountered several technical issues of using the platform. We made the platform and the exercises in one VBox image. Students can download the VBox image and run it in their own computers. However, a few students are

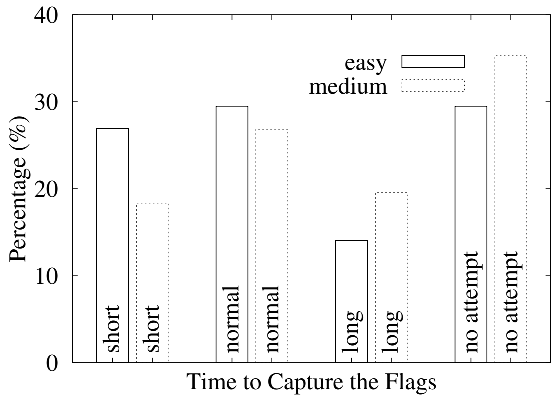

Figure 5: Comparison of Easy and Medium Exercises

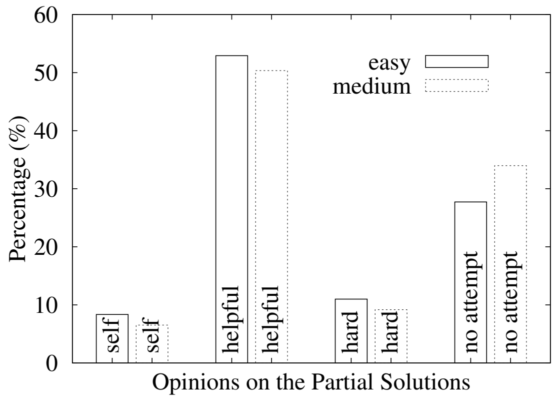

Figure 6: Helpfulness of Partial Solutions

using tablet computers that do not support Virtual Box. Another issue is that many students did not have enough computer administration skills to install and setup the needed tools and libraries in their own computers. In particular, many tools and libraries are Linux oriented. The students who use Windows and Mac OS X struggled to get the tools installed and working. About 13% of students gave up on the exercises due to these issues. This new factor was not discussed in the literature that beginners can be dissuaded due to the difficulty of installing and using security tools.

Difficulty of Exercises: As discussed in Section 3.3, the CTF challenges have different levels of difficulty. In our study, a half of the assigned CTF exercises were at the easy level, and the other half were at the medium level. We asked the students to report the time they spent on getting the flags in the survey. The results are shown in Figure 5. "Short" means the time fewer than 15 minutes, "medium" means the time between 15 and 40 minutes, "long" means the time greater than 40 minutes, and "no attempt" means the students did not work on the exercises.

We can clearly observe the difference between the easy exercises and the medium ones. More students spent longer time on the medium ones and more students did not attempt the medium ones. Meanwhile, more than two thirds of students could get the flags in both types of exercises. One third of students did not attempt because they encountered the aforementioned technical issues or were not very motivated to take further exercises beyond textbook questions. The exercises are overall suitable to most beginners, since they are challenging but solvable to beginners.

Helpfulness of Partial Solutions: We provided the partial solutions of the exercises and asked the students if the solutions were helpful in the survey. The results are shown in Figure 6. "Self" means the students got the flags without reading the solutions, "helpful" means the students got the flags with the help of the solutions, "hard" means the students did not understand the solutions, and "no attempt" means the students did not work on the exercises.

As expected, only a few students (less than 8%) could get the flags by themselves. Most students were beginners and needed to read the solutions regardless the difficulty of the exercises. Because only about 10% of students could not understand the solutions, we think the solutions are suitable for beginners to study too. Overall, 83% of students felt the CTF exercises truly helped them to understand computer security.

6 Conclusion and Future Works

In this paper, we analyzed the solutions of about 3600 CTF challenges from 160 security competitions in the past three years. We identified major security issues, skills, and techniques that are needed for beginners to participate in the competitions. After comparing different options, we built a CTF platform based on PicoCTF for the beginners and designed exercises in six categories: coding, cryptography, reverse engineering, pwn, forensic and web. We provided the platform and the exercises in our introductory computer security course. The feedback from the students showed positive on the usage of this package in the course.

The CTF exercises do not address all aspects of security education though. First, the exercises are mainly focused on offensive techniques. Defensive techniques and skills are missing in this study, and they are another indispensable part of computer security. Second, the exercises do not include any practices on system administration and management, which are important to computer security practice too. In future, we will explore new approaches to involve beginners on these two aspects.

References

[1] CTF Time. https://ctftime.org/.

[2] CTF Write-ups. https://github.com/ctfs.

[3] CTFd. https://github.com/CTFd/CTFd.

[4] FbCTF. https://github.com/facebook/fbctf.

[5] OpenCTF. https://github.com/EasyCTF/OpenCTF.

[6] PicoCTF. https://github.com/picoCTF.

[7] TinyCTF. https://github.com/balidani/tinyctf-platform.

[8] Jonathan Burket, Peter Chapman, Tim Becker, Christopher Ganas, and David Brumley. Automatic Problem Generation for Capture-the-Flag Competitions. In *Proc. of USENIX Summit on Gaming, Games, and Gamification in Security Education*, 2015.

[9] Martin Carlisle, Michael Chiaramonte, and David Caswell. Using CTFs for an Undergraduate Cyber Education. In *Proc. of USENIX Summit on Gaming, Games, and Gamification in Security Education*, 2015.

[10] Peter Chapman, Jonathan Burket, and David Brumley. PicoCTF: A Game-Based Computer Security Competition for High School Students. In *Proc. of USENIX Summit on Gaming, Games, and Gamification in Security Education*, August 2014.

[11] Tom Chothia and Chris Novakovic. An Offline Capture The Flag-Style Virtual Machine and an Assessment of Its Value for Cybersecurity Education. In *Proc. USENIX Summit on Gaming, Games, and Gamification in Security Education*, 2015.

[12] Kevin Chung and Julian Cohen. Learning Obstacles in the Capture The Flag Model. In *Proc. of USENIX Summit on Gaming, Games, and Gamification in Security Education*, August 2014.

[13] Adrian Dabrowski, Markus Kammerstetter, Eduard Thamm, Edgar Weippl, and Wolfgang Kastner. Leveraging Competitive Gamification for Sustainable Fun and Profit in Security Education. In *Proc. of USENIX Summit on Gaming, Games, and Gamification in Security Education*, 2015.

[14] Andy Davis, Tim Leek, Michael Zhivich, Kyle Gwinnup, and William Leonard. The Fun and Future of CTF. In *Proc. of USENIX Summit on Gaming, Games, and Gamification in Security Education*, August 2014.

[15] Jelena Mirkovic, Aimee Tabor, Simon Woo, and Portia Pusey. Engaging Novices in Cybersecurity Competitions: A Vision and Lessons Learned at ACM Tapia 2015. In *Proc. of USENIX Summit on Gaming, Games, and Gamification in Security Education*, 2015.

[16] Z. Cliffe Schreuders and Emlyn Butterfield. Gamification for Teaching and Learning Computer Security in Higher Education. In *Proc. of USENIX Workshop on Advances in Security Education*, 2016.

[17] Giovanni Vigna, Kevin Borgolte, Jacopo Corbetta, Adam Doupé, Yanick Fratantonio, Luca Invernizzi, Dhilung Kirat, and Yan Shoshitaishvili. Ten Years of iCTF: The Good, The Bad, and The Ugly. In *Proc. of USENIX Summit on Gaming, Games, and Gamification in Security Education*, 2014.

[18] Jan Vykopal and Miloš Barták. On the Design of Security Games: From Frustrating to Engaging Learning. In *Proc. of USENIX Workshop on Advances in Security Education*, 2016.