Defence CTF Platform "Defender"

Konstantin Veselov
kaveselov@edu.hse.ru
Higher School of Economics
Moscow, Russia

Abstract

Capture The Flag (CTF) security competitions are now an important part of cybersecurity education, used globally to a great extent. While they are becoming more popular and competitive, for the second most popular format – attack-defence, there is no way to train other than playing it, and some parts of it (e.g. server administration, hosting a flag farm, running exploits) could be a barrier to entry for a lot of participants. This paper presents a new way of training and introducing new people to the concepts of attack-defence CTF, which can be hosted without time and latency limits, require less server administration skills, and can be distributed across different computers to allow a high number of participants. It is achieved through automation of running exploits, teacher-provided exploits to check the defence of the participant, and running services in separate virtual machines, so participants cannot see how they are attacked

Keywords: Capture The Flag; CTF; Attack-Defence; Distributed systems; Virtual Machines; Cybersecurity; Gamification

1 Introduction

In a cybersecurity context, Capture The Flag is a general term for a kind of competition, where the main goal of participants is to get some secret data, which is called flags from either the event organizers or the other event participants using computers or hacking-related means.

There are two dominant competition types - jeopardy or task-based, and attack-defence (according to ctftime.org statistics accessible in [15]). Here, we will focus on the latter. In Attack-defence CTF, all participating teams are given an identical machine with several vulnerable services and are required to defend their services and attack the services of other teams. To keep track of the attacks, in each service there is a way to store some secret data (flags), and in every game round (usually a few minutes), organizers of the competition generate flags and store them in the participants' services. To register an attack, a participant must get flags from the other team, and show them to the organizers. [20] Modern attack-defence CTFs send flags automatically and expect participating teams to have a system to automate attacks, that sends flags back to the organizers to receive a reward. Also, a common and useful tactic is to monitor all traffic that goes to your server and try to replicate the attack on other teams. This is extremely important because it sometimes enables teams that are being exploited to make their exploits and catch up to the team, that originally developed the exploit.

CTFs are now a common tool to teach cybersecurity [3, 6, 7, 9, 16], for example in 2024, one of the tracks of Russian National Olympiad in technology is security, and the competition is partly a CTF [21]. CTF can be a way to gamify security education, which can lead to a more engaging experience [9, 17], and improve the self-confidence of students [9].

Playing attack-defence CTFs can be difficult for beginners because it requires preliminary administration skills to host a system of automating attacks, and figuring out how to get traffic from their machine in an efficient way (usually, a special service for sniffing and displaying live traffic is used). Also, according to ctftime.org statistics from [15], there are a lot less attack-defence competitions in general, which makes training by playing more difficult.

Hosting an attack-defence training can also be difficult, as the organizers are required to have networking and administering skills, and a minimal sensible amount of people playing it needs to be around 10 (2 teams) [19].

The goal of this project is to create more options for training for Attack-Defence CTFs, increasing their availability for beginners, and making more long-lasting competitions for advanced players. It will achieve this by removing the time-based element of the competition, and grade solutions independent of their timings. This should also relieve some stress of the participants, allowing them to figure out the best way for them to administer their systems, and advance their skills in information security.

The article is structured as follows. Similar systems and present training methods are described in Section

2. The Defence CTF format is described in Section 3. Implementation details are described in Section 4 The expected results of this project are discussed in Section 5. Section 6 concludes the paper.

2 Present training methods

The [19] report covers a wide range of attack-defence CTF issues, including those not covered in this article. My implementation generally follows a solution proposed in this report, in that it runs services in virtual machines to ensure simplicity of administering. However, the report mainly focuses on making competitive events fair, and mitigating cheating, while my project focuses on training. The report also covers some possible issues of hosting a lot of virtual machines, which can affect the latency and usability of the system, and although it is relevant in the case of competitions, in training high latency will not be as critical.

Several training resources run CTF with no time limit, such as picoGym [12], but they host jeopardy CTF and not attack-defence.

To date, several competitions with similar alterations to the attack-defence format have been hosted, although little written record exists about them. School competition QCTF Starter 2018.2, as a part of its jeopardy tasks, contained one task of defending a service in an attack-defence style, but it was criticized for being highly unstable, as described in a blog post [8].

3 Defence CTF

This paper introduces a novel CTF training format, similar to that described in [19] talk. This section describes the format in detail.

Defence CTF consists of two stages. The first stage is attacking and defending against a predefined set of exploits that is prepared by the organizers. The second or the scoreboard stage allows the participant to submit attacks on solutions of other participants, and update their defences, according to the state of their solution of a general scoreboard. The following sections explain these stages in detail.

3.1 General site composition

At the main site page, the participant is required to authenticate in this system, as running virtual machines and exploits is a computationally expensive task. Then, the participant is greeted with a list of tasks. After selecting one of the tasks, they proceed to the first stage.

3.2 First stage

When the participant enters the first stage, they are presented with an explanation of the format, and a request to perform an attack. They do it by investigating the source code of the given task and using a demonstration virtual machine hosting this task without any defences. After they find a vulnerability, they are required to make an automated attack in the format, compatible with most attack automation systems. This is designed this way to ensure that the ability to make automated attacks on a training platform can be converted to an ability to make automated attacks in a real attack-defence competition. Then, the participant submits this attack to the system, which checks it against a newly created virtual machine with this task. If the attack works correctly and returns stolen flags, the participant proceeds to the next section - defence.

The defence stage gives the participant full control of a newly created virtual machine, running this task. The access to the machine is given through SSH protocol [18]. The participant could submit a patch to the found and exploited vulnerability, and then tell the system to check the virtual machine for vulnerabilities. The system checks first checks that the service is working correctly, then tries to exploit the prepared vulnerabilities. It is important that in this stage, the participant could see all of the traffic that the system sends them. This emulates the real experience of defending in an attack-defence game, as the participant could prepare the traffic sniffing infrastructure in the same way.

3.3 Scoreboard stage

After completing the defence section, the participant could proceed to play against other participants in the scoreboard stage. It is designed to encourage competition and ensure that participants defend thoroughly and correctly.

In the scoreboard stage, all the participants that completed the first stage could attack each other, and deploy new versions of their defences.

The more points a participant has, the higher they are on a scoreboard. Total points are calculated this way:

$$\frac{\#\ teams\ that\ you\ attack}{\#\ of\ teams}+(1-\frac{\#\ teams\ that\ attack\ you}{\#\ of\ teams})$$

As the participant could defend against a static organizer-prepared exploit using a signature from the exploit traffic, it is important to allow participants to attack each other using custom, more robust exploits. To prevent dubious defences, the system also does not allow the participant to see the exploit traffic at this stage.

4 Implementation details

The system is divided into 4 components with the ability to scale individually. The 4 components are

1. **Frontend**. It consists of ReactJS [13] single page application, that calls API through HTTP protocol. It is served through an nginx [14] web server.
2. **API**. This component is written in Python, using FastAPI, Pydantic, and a PostgreSQL database.
3. **Machine workers**. This component is written in Golang, using libvirt [2] for running virtual machines, and QCOW2 format [10] for storing memory of images and snapshots. To download or upload a QCOW2 image, the machine worker uses S3 file storage.
4. **Celery exploit and checker runners**. To run exploits and checks against virtual machines, API stores the exploit code in the S3 storage and asynchronously calls a special exploit or checker runner using Celery worker queue [1]

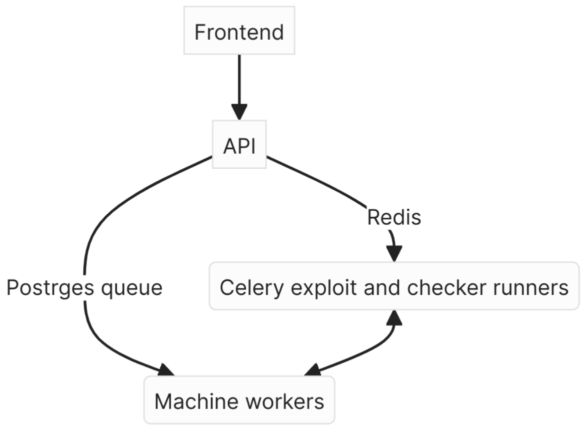

Fig. 1. Components of the system

4.1 Frontend

The frontend section should be simple, although scalable and reactive. ReactJS was chosen for its widespread adoption, and availability of tutorials, which could simplify the development. It connects to the API through HTTP protocol, using JSON for data transfer.

4.2 API

API is a FastAPI server, that uses MVC design. Models use Pydandic classes for type-checking. Views are HTTP handlers, that use Pydantic models to check the validity of user input. Also, there is a separate package for database interaction, to encapsulate and separate the business and the database logic. API has 3 controller packages: exploits, defences, and scoreboard.

4.3 Machine workers

Machine workers are designed to have great performance and security, thus Golang is used. The system is required to allow dynamically adding new machine workers to adjust for increased load, so the queue architecture should also allow it. It uses PostgreSQL for the queue implementation, because it allows for exactly-once delivery of the messages, compared to Kafka and RabbitMQ which only grant at-least-once delivery guarantee [4]. Also, using Postgres, the system could transactionally claim that the work is done, and in the same transaction set the results of this work.

Machine worker code uses a repository pattern [11] to encapsulate several implementations:

The machine worker periodically polls PostgreSQL for new work, claims it, does it, and marks it as done. It is similar to the distributed inbox pattern [4], but with a worker assignment part.

4.4 Celery workers

Some business logic of running exploits and checkers is separated into independent celery functions. Celery allows to run workers on several machines, using Redis to synchronize, allowing horizontal scaling.

5 Expected results

The intended outcome of this program is to allow more students to play attack-defence CTFs, improve their skills in writing exploits, and in defending their services in attack-defence environments. Design decisions, described in this paper, should allow this system to scale horizontally, allowing a lot of participants to access this system. But this system should also work in a small-scale environment, where it is hard to involve a lot of students to attend a normal attack-defence training, even if the teacher has the skills to host one.

Overall, this project should popularize the attack-defence format, attracting more people to the cybersecurity community, and allowing more people to learn how to properly defend their systems. It will also help teachers to prepare their students for attending an attack-defence CTF event.

6 Conclusion

With more interest in cybersecurity and CTF globally, the importance of education is undeniable. Discovering more ways to educate students should be beneficial for everyone, as this opens new opportunities for some people, which is crucial in improving education systems at all levels. As cybersecurity is known to have a sizeable shortage of workers, it is very important to recruit a young and diverse audience to pursue education and careers in cybersecurity, according to [5]. This said future work opportunities would even more interest students in playing CTFs and using this project.

Attack-defence CTF is one of the foundational formats of the CTF history. However, the fact that it is less available to the novice student audience hinders its ability to be as widespread as it can be. My project aims to make it more available and provide all students with an opportunity to learn how to play in this format.

References

[1] "Celery task queue documentation." [Online]. Available: https://docs.celeryq.dev/en/stable/

[2] M. Bolte, M. Sievers, G. Birkenheuer, O. Niehörster, and A. Brinkmann, "Non-intrusive virtualization management using libvirt," in *2010 design, automation & test in europe conference & exhibition (date 2010)*. IEEE, 2010, pp. 574–579.

[3] L. K. Chen, M. H. Jenalis, and J. Juremi, "Towards inclusive cybersecurity learning: A novice-friendly capture-the-flag onboarding platform," *Journal of Applied Technology and Innovation (e-ISSN: 2600-7304)*, vol. 7, no. 4, p. 60, 2023.

[4] O. Dudycz, "Outbox, inbox patterns and delivery guarantees explained," 2020. [Online]. Available: https://event-driven.io/en/outbox_inbox_patterns_and_delivery_guarantees_explained/

[5] M. H. Dunn and L. D. Merkle, "Assessing the impact of a national cybersecurity competition on students' career interests," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 2018, pp. 62–67.

[6] Z. Kaplan, N. Zhang, and S. V. Cole, "A capture the flag (ctf) platform and exercises for an intro to computer security class," in *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 2*, 2022, pp. 597–598.

[7] M. Knüpfer, T. Bierwirth, L. Stiemert, M. Schopp, S. Seeber, D. Pöhn, and P. Hillmann, "Cyber taxi: A taxonomy of interactive cyber training and education systems," in *Model-driven Simulation and Training Environments for Cybersecurity: Second International Workshop, MSTEC 2020, Guildford, UK, September 14–18, 2020, Revised Selected Papers 2*. Springer, 2020, pp. 3–21.

[8] V. Lebedev, "Qctf starter 2018.2 — разбор," 2018. [Online]. Available: https://medium.com/@d3fl4t3/qctf-starter-2018-2-%D1%80%D0%B0%D0%B7%D0%B1%D0%BE%D1%80-c30da1d22720

[9] K. Leune and S. J. Petrilli Jr, "Using capture-the-flag to enhance the effectiveness of cybersecurity education," in *Proceedings of the 18th annual conference on information technology education*, 2017, pp. 47–52.

[10] M. McLoughlin, "The qcow2 image format." [Online]. Available: https://raw.githubusercontent.com/zchee/go-qcow2/master/docs/specification.md

[11] F. Pereira, E. Guerra, and R. R. Rosa, "Patterns for polyglot persistence layer," in *Proceedings of the 29th Conference on Pattern Languages of Programs*, 2022, pp. 1–8.

[12] picoCTF, "picogym practice challenges." [Online]. Available: https://play.picoctf.org/practice

[13] P. Rawat and A. N. Mahajan, "Reactjs: A modern web development framework," *International Journal of Innovative Science and Research Technology*, vol. 5, no. 11, pp. 698–702, 2020.

[14] W. Reese, "Nginx: the high-performance web server and reverse proxy," *Linux Journal*, vol. 2008, no. 173, p. 2, 2008.

[15] V. Švábenskỳ, P. Čeleda, J. Vykopal, and S. Brišáková, "Cybersecurity knowledge and skills taught in capture the flag challenges," *Computers & Security*, vol. 102, p. 102154, 2021.

[16] J. Vykopal, V. Švábenskỳ, and E.-C. Chang, "Benefits and pitfalls of using capture the flag games in university courses," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 2020, pp. 752–758.

[17] B. Wolfenden, "Gamification as a winning cyber security strategy," *Computer Fraud & Security*, vol. 2019, no. 5, pp. 9–12, 2019.

[18] T. Ylonen and C. Lonvick, "The secure shell (ssh) protocol architecture," Tech. Rep., 2006.

[19] А. Гейн, "Ructf 2017. Почему классический ctf должен умереть," 2018. [Online]. Available: https://www.youtube.com/watch?v=AnGx8iLLNuI

[20] И. Новиков, Н. Ткаченко, "Attack-defense ctf. Что это такое и как в это играть." [Online]. Available: https://cbsctf.ru/ad

[21] МИЭМ, "ВсОШ по направлению Информационная безопасность предмета Технология 2023/2024," 2023/2024. [Online]. Available: http://vsosh.miem.hse.ru/

Word count: 1963