This article has been accepted for publication in the Journal of Network and Computer Applications:
Kucek, S., Leitner, M., 2020. An Empirical Survey of Functions and Configurations of Open-Source Capture the Flag (CTF) Environments. Journal of Network and Computer Applications 151, 102470.
https://doi.org/10.1016/j.jnca.2019.102470

An Empirical Survey of Functions and Configurations of Open-Source Capture the Flag (CTF) Environments

Stela Kucek[a], Maria Leitner[a,*]

[a] *AIT Austrian Institute of Technology, Center for Digital Safety & Security, Giefinggasse 4, 1210 Vienna, Austria*

Abstract

Capture the Flag (CTF) is a computer security competition that is generally used to give participants experience in securing (virutal) machines and responding to cyber attacks. CTF contests have been getting larger and are receiving many participants every year (e.g., DEFCON, NYU-CSAW). CTF competitions are typically hosted in virtual environments, specifically set up to fulfill the goals and scenarios of the CTF. This article investigates the underlying infrastructures and CTF environments, specifically open-source CTF environments. A systematic review is conducted to assess functionality and game configuration in CTF environments where the source code is available on the web (i.e., open-source software). In particular, from out of 28 CTF platforms, we found 12 open-source CTF environments. As four platforms were not installable for several reasons, we finally examined 8 open-source CTF environments (*PicoCTF, FacebookCTF, HackTheArch, WrathCTF, Pedagogic-CTF, RootTheBox, CTFd and Mellivora*) regarding their features and functions for hosting CTFs (e.g., scoring, statistics or supported challenge types) and providing game configurations (e.g., multiple flags, points, hint penalities). Surprisingly, while many platforms provide similar base functionality, game configurations between the platforms varied strongly. For example, hint penalty, time frames for solving challenges, limited number of attempts or dependencies between challenges are game options that might be relevant for potential CTF organizers and for choosing a technology. This article contributes to the general understanding of CTF software configurations and technology design and implementation. Potential CTF organizers and participants may use this as a reference for challenge configurations and technology utilization. Based on our analysis, we would like to further review also commercial and other platforms in order to establish a golden standard for CTF environments and further contribute to the better understanding of CTF design and development.

Keywords: CTF, capture the flag, cyber range, computer network operations, cyber security exercises, cyber security training

1. Introduction

Capture the Flag (CTF) is a challenge-based competition for gaining and training cyber security related skills by actively applying them. In a CTF, a team (or a single player) solves problems related to cyber security, and if their answer (also often referred to as flag or solution) to the problem is correct, they get rewarded with points. The aim is to score more points than other participants, which contributes to the competition. CTF platforms (also often referred to as CTF environments or CTF frameworks in literature) are typically used as game-like environments where participants may practice computer security abilities, skills and knowledge. Various attack techniques or skills can be tested such as (e.g., [1, 2, 3]). Long-term goals of CTF competitions e.g., in education would be to allow participants to test computer security skills and abilities, enhance hands-on experience and challenge and test the participants (see e.g., [4, 5]). A more elaborate definition of CTFs and other terms can be found in Section 2.1.

In the past years, many CTF competitions have been conducted and attracted many participants

*Corresponding author.
Email addresses: stela.kucek.fl@ait.ac.at (Stela Kucek), maria.leitner@ait.ac.at (Maria Leitner)

(e.g., DEFCON[1], MITRE CTF[2], NYU-CSAW [6], iCTF [4]). CTF competitions are typically hosted in virtual environments, specifically set up to fulfill the goals and scenarios of the CTF game. While many CTF competitions are assessed regarding the learning obstacles (e.g., [7, 8]), view of participants (e.g., [5]), evaluation theory ([9]) and design of cyber security exercises (e.g., [6, 10, 11, 12]). We found only single instances where software environments of CTFs are described such as [4, 13, 14]. In addition, we could not find any research that focuses on a systematic assessment of the configuration and setup of virtual environments for live security competitions such as CTFs.

This paper aims to fill this gap and systematically assesses and reviews technical infrastructures[3] for CTF environments. This article adresses three research questions (RQ):

RQ1 What are typical software environments for CTF competitions?

RQ2 Which design features and functionality does CTF software have?

RQ3 Which challenge types are supported in CTF software?

RQ4 How can CTF games be configured in CTF software?

RQ1 focuses to evaluate the current landscape of software environments for CTF competitions while *RQ2*, *RQ3* and *RQ4* center on functionality of CTF environments, challenge types and game configuration in CTF software.

To address these questions, an empirical [15] review is conducted on software environments for CTFs. This article aims to analyze and evaluate empirically CTF technologies based on observations, implementation of concrete examples, and systematic testing of functionality.

The methodology of this review consists of 5 steps: (1) review of CTF environments and software, (2) CTF platform selection, (3) installation of open-source CTF platforms, (4) challenge development and implementation, (5) analysis and evaluation of CTF platforms. Out of 28 CTF platforms[4], we found 12 "open-source" CTF environments. Due to the limitations of reviewing proprietary software in depth, we chose to select only open source software due to the advantages to review the source code. In this article, we use the term *open-source* here for software that has an open-source license (e.g., GPL, MIT, etc.) or has no attribution at all. Out of the 12 open-source platforms, 4 appeared to have significant bugs or insufficient documentation and support and were therefore not taken into further account. Finally, 8 open-source CTF platforms were analyzed regarding their features and functions for hosting exercises and providing a challenge configuration.

Based on this review, we identify existing terms and terminology in CTFs (cf. Section 2.1). Another result is an overview of common software architectures and technologies in CTF platforms (cf. Section 4.2). The individual analysis of the platforms (cf. Section 4.3) provides insight into features of open-source CTF platforms (cf. Section 4.3.1) and the CTF game design and configuration options (cf. Sections 4.3.2 and 4.3.3). The findings of this article contribute to the

- understanding of engineering CTF software platforms,
- design of game configurations, and
- utilization and variability between CTF platforms.

The results of this article can be used by different audiences:

- Current or prospective **CTF organizers** or **educators** may use the results to assess, compare and evaluate CTF environments. They may select or choose differently when knowing the selection criteria and variability between configurations.
- **CTF developers** or **designers** may use this systematic analysis of open-source CTF frameworks to improve current or future CTF frameworks. As the article reviews features, new functions and configurations might be added to CTF environments.
- Furthermore, it may help **CTF participants** understand the diversity of scoring mechanisms and challenge options.

[1] `https://defcon.org/`

[2] `https://mitrestemctf.org/`

[3] This article focuses only on related software and not on any hardware-related choices.

[4] In this paper, we use the term "*CTF platforms*" synonymously for CTF software, environments, frameworks or similar.

- Lastly, others **computer security researchers** may use this survey as a basis and reference point for further analysis and research on CTF assessment, design and development.

This article is structured as follows: Section 2 reviews definitions, delivery models of the CTF environments and background on CTFs. Section 3 describes the methodology of the survey and the performed steps in detail. Section 4 provides information on the results of this survey, including: an overview of the examined CTF environments, common activities supported by these platforms, developed CTF challenges, and finally the individual analysis of the platforms in terms of supported features, challenge types and challenge configuration options. Furthermore, Section 5 discusses the main findings of the survey, impact on research and practice and limitations of the survey. Section 6 concludes the article.

2. Background

This section summarizes the basic definitions for CTF environments and reviews background and related work related to the topic of the article.

2.1. Overview

Capture the flag (CTF) is a computer security competition with challenging exercises from different categories (e.g., network security, application security, mobile security, etc.). Typically, the goal of this game is to find a *flag*, i.e. the solution to the given problem (which is often solved through multiple steps or tasks within an exercise). A *flag* can be, for example, a string in the format `flag{sometext}` but also an image or another element defined by the CTF game designers. Table 1 displays a list of definitions found in literature. Within literature, we discovered several CTF games or challenge types. The definitions can be found in Table 1 and are briefly summarized in the following:

1. ***Quiz (Q)*** consists typically of a question-answer pair from the computer security or related domains.
2. ***Jeopardy (J)*** is a sequence of challenges or tasks which require computer networking knowledge and skills by the participants, and the goal is to solve these tasks in order to get to the flag.
3. ***Attack-defense (A)*** supports offensive and defensive hacking between teams.
4. ***Mixtures (M)*** are typically a combination of jeopardy and attack-defense CTF.
5. ***King of the Hill (K)***: the main goal is capturing a target system and keeping it under control for as long as possible. A *king of the hill* challenge is characterized by a vulnerable system that is to be captured. It is a form of penetration testing, which is defined in [16] as *"simulation of an attack on a system, network, piece of equipment or other facility, with the objective of proving how vulnerable that system or 'target' would be to a real attack."*

`CTF platforms` (also often referred to as CTF environments or CTF frameworks in literature) are typically used as game-like environments where participants may practice computer security abilities, skills and knowledge. They are mostly used to manage the logistics of keeping track of players and flags within the games. Hence, we use the term `CTF platforms` in this article as software that is often concerned with the organization (e.g., teams, flags) of CTFs.

Typically, CTF competitions are hosted in virtualized environments. Hence in addition to CTF platforms that host the logistics, the administrators of CTFs still have to create and manage virtual machines (e.g., running attackable services depending on the challenges). In computing, virtualization refers to the abstraction of a physical component into a logical object, which allows for higher degree of utilization of the resource that the object provides (e.g. virtual LANs provide greater network performance and improved manageability by being separated from the physical hardware). Essentially, virtualization is the ability to abstract a physical server into a virtual machine [25].

CTFs have been conducted in virtual test beds, virtual environments, clouds, cyber ranges or similar environments (see Table 2). For example, the definition of cyber ranges (CR) have emerged, particularly in the military context, and are virtual environments for practicing computer networking and other related skills such as penetration testing, defending networks, hardening critical infrastructure and responding to attacks [14]. Hence, many different names exist for virtual environments that may host CTFs.

Table 1: CTF terms and definitions

Capture the Flag (CTF)	**Ref.**
"CTF, which stands for 'Capture the Flag', is a computer-based competition used to teach information security skills through hands-on experience. Players earn points and gain control of the game map by completing tasks with their team that identify potential threats, and secure computers and networks against virtual attacks. Because learning attack techniques is often the best way to learn how to protect against them, CTFs teach both defense and offense skills. Points are awarded to teams on the basis of speed and accuracy. At the end of the allotted time, the team with the most points is declared the winner."	[17]
Quiz	
"Quizzes are the most basic type of Level, allowing for a simple question and answer format. They are not case sensitive."	[17]
Jeopardy	
"Jeopardy CTFs are the most common kind of CTF. They revolve around a set of challenges which are provided by competition organizers to competitors. Competitors form teams and then work on the challenges together. Each challenge is designed so that when the competitor solves it, a small piece of text or "flag" is revealed. The flag is then submitted to a website or scoring engine in exchange for points. The amount of points rewarded is typically relative to the perceived difficulty of the challenge."	[18]
"Jeopardy-style CTF games have the organizers running a set of challenges that each individual/team has to solve for points. In general, the more complicated the task the more points earned. Challenges are generally independent from each other and ideally idempotent between connecting players, which leads to reliability and stability for large contests. Scoring is nice and simple too: add up the points for solved challenges, and use timing of solutions to break ties. Categories may include Web, Forensics, Crypto, Binary, and Lock Sport Challenges."	[19]
Attack-Defense	
"In an attack-defense CTF, teams attack each other in a special network cut off from the outside world."	[20]
"Here every team has own network(or only one host) with vulnerable services. Your team has time for patching your services and developing exploits usually. So, then organizers connects participants of competition and the wargame starts! You should protect own services for defense points and hack opponents for attack points.	[21]
King of the Hill	
"Bases are a specialized type of Level, representing a target system which must be compromised by teams in order to capture points. This is used in a King of the Hill game type, ***where teams compete over control of these target systems"****.*	[17]
"King of the Hill (KOTH) is a type of cybersecurity competition in which ***multiple teams fight for control of a large network. Each team is given machines to defend, and must attack other team's boxes."***	[22]
*"NetKotH (Network King of the Hill) is a type of Capture the flag (CTF) that is much easier to build and administer than traditional CTFs.****NetKotH consists of a network of one or more computers running virtual machines containing the challenges****. The number of challenges is up to the NetKotH host. The ScoreBot (scoring engine) can be a standalone computer or virtual machine running a web server and the scoring software. The scores can be viewed by any computer on the NetKotH network using any web browser.* ***The contestants must gain access to a challenge machine which may contain multiple vulnerabilities****. Once they are on the challenge machine, they must plant their team tag where the ScoreBot can see it."*	[23]
"The aim of the participants of King of the Hill is to detect vulnerabilities of the systems, exploit them and, the most important of all, keep control over the systems as long as it is possible. The trick is in regeneration of the sets of vulnerabilities in the systems."	[24]

2.2. Delivery Models

CTF environments support various delivery models that allow e.g., CTF access 24/7 or just once in a month. In particular, we found several delivery models for CTFs: as hosting service, live competition, online competition, online training, or as a local installation (see Table 2).

Please note that there might be other CTF delivery models that are not included in this section, however, we found that these were the most common delivery models used in 2018 (see Section 5.4). The rest of this article focuses only on the local installation as delivery model.

2.3. Related Work

Research discovers and describes beneficial effects of learning cyber security key concepts through challenges and problems embedded in competition environments. For example in [26], state-of-the-art simulation systems for information security education, training and awareness are reviewed and the authors state that the "weakest link" in the implementation of security policies are people, and that therefore this link needs to be strengthened by providing better education, appealing practical training, and raising general awareness on information security concerns. Similarly, Gavas et al. [6] focus on cyber security challenges, their impact, purpose and value. The authors explain why education in information security is important and why a competition-based learning approach is useful and effective. The benefits of the "gaming" aspect of a CTF environment are described by Boopathi et. al [8]. The authors describe the gaming approach of the components of InCTF (Indian Capture the Flag). In contrast to emphasizing the benefits of challenge based learning in cyber security (e.g., [27]), Chung and Cohen [7] bring forward a critical view of the CTF learning model: they emphasize issues related to participation, quality assurance, and confusing challenges, all of which ultimately affect the overall quality of a CTF competition. In this context they present insights and lessons learned from organizing CSAW CTF, their online CTF competition. In summary, the publications describe CTF competitions as an effective tool for computer security training and that the gaming approach will support increase the number of trainees (i.e. students) in computer security, thereby securing the "digital world".

Furthermore, research provides various methods (e.g., CTF competitions or similar) of delivery for cyber security education. A conceptual analysis of cyber security education is presented by Katsatonis et. al [28], where the authors use live competitions as a basis for evaluation. Abwajy [29] provides the results of a study on user preferences regarding cyber security awareness delivery methods. Having tested text-, game- and video-based delivery methods in terms of user-preference, the author's study suggests that combined delivery methods are more efficient than individual security awareness delivery methods. Davis and Magrath [14] conduct a survey of CRs and testbeds, reviewing the newest CRs and related testbeds in 2013. They aim to describe the purpose and functionality of at that point newly published examples in order to provide assistance to organizations which are deciding on implementing or using a CR. While they focus on a survey of testbeds in general, this article focuses on the analysis and game design in CTF software. Similarly, Khoo [30] describes the design, development and validation of a CTF cyber security education framework as an experiment to study the relationship between the learners' motivation and achievement level. This article does not conduct assessments on the motivational and achievement aspects of students using CTF platforms, but analyzed CTF software in terms of functionality. Many more examples exist that focus on teaching cyber security aspects. For example, Bock et al. [31] utilize king of the hill examples for teaching penetration testing or Wi et al. [32] use a git-based CTF for attack-defense competitions.

Raman et al. [33] propose a framework for evaluation and ranking of CTFs according to factors such as similarity of the tasks to the common critical vulnerabilities, solvability of tasks, periodicity, training given prior to CTF, and geographical reach. The authors exercised their framework to compare five CTF contests: *DEFCON*, *SECUINSIDE*, *CSAW*, *Mozilla CTF* and *PHD CTF*. In this article, we evaluate open-source software from a rather technical aspect (features and challenge configuration possibilities) and do not focus on the semantic aspects of CTF competitions in general (e.g., contents and relations). In this article, we use content samples (our examples see Section Appendix B in the appendix) in order to conduct our assessment. Chung [34] describes CTF competitions and compares CTFd briefly to FBCTF, OpenCTF, picoCTF, TinyCTF, Mellivora, and the iCTF framework (cmp. list of platforms in Section 4.1). This article provides a more extensive

Table 2: CTF delivery models

Delivery Model	Description	Examples
Hosting Service	Platform is used for an arranged type of CTF competition, provided as a service	*CTFd*, *ShellWePlayAGame*
Live Competition	On-the-spot participation in a CTF competition	*DEFCON CTF*, *MITRE CTF*, *MIT Lincoln Laboratory CTF*, *RootTheBox*, *HackTheArch*
Online Competition	Remote, online participation in a CTF competition	*EasyCTF*, *Google CTF*, *Hacking Lab*, *PicoCTF*
Online Training	Remote, online participation in a CTF training	*CTF365*, *The Hacking Dojo*
Local Installation	Source code of the platform is publicly available and can be installed locally - any form of CTF event can be realized	*CTFd*, *FBCTF*, *PicoCTF*, *RootTheBox*, *Mellivora*

evaluation using also content samples.

Noor et al. [35] evaluate the usability of open-source and online CTF platforms in a report. The review of this article was conducted a bit earlier but at a similar time and focused on the assessment of the usability of online CTF platforms. This article focuses on the features and configuration options that CTF platforms offer for users.

To the best of our knowledge, no survey on CTF environments, their features and challenge configuration options has been published. Therefore, this article aims to fill this gap and to assess the design and development of CTF environments as well as challenge configurations.

3. Methodology

This survey was conducted by performing five steps (see Figure 1). Each step shown in Figure 1 is outlined in the following sections.

3.1. Review of CTF Environments and Software

First, a review of existing CTF software (e.g., open-source, commercial, free) and general aspects concerning the realization and implementation on cyber security training was conducted. The goal was to gather information on existing CTF software on the Internet as well as related literature. The search was carried out using online search engines such as *Google*, *Google Scholar* (`http://scholar.google.com`) and *IEEE Xplore Digital Library* (`http://ieeexplore.ieee.org`). The keywords used during the searches were e.g., *"cyber security exercises"* as well as *"cyber security training"* and *"capture the flag platform"*. Unfortunately, the term "CTF" generated too many generic results therefore we could use it only in combination with the aforementioned keywords. We have tested the review with related terms and synonyms in the same search engines which produced less or similar search results. Note that this search focused primarily on software that describes itself as a tool for CTF competitions or similar events. The review resulted in a list of 28 CTF platforms that are described in the appendix in Table A.11. Note that the research and evaluation of these open-source platforms was performed in the between August and December 2017 and that since then there could have been changes and further developments. The results of this study reflect that time span.

3.2. CTF Platform Selection

Based on the list of 28 platforms (see Table A.11 in the appendix), we identified 12 platforms as "*open-source*" software. Open-source software is software with publicly accessible source code, i.e. anyone can inspect, modify, and enhance the code. In this article, we use the term "*open-source*" as

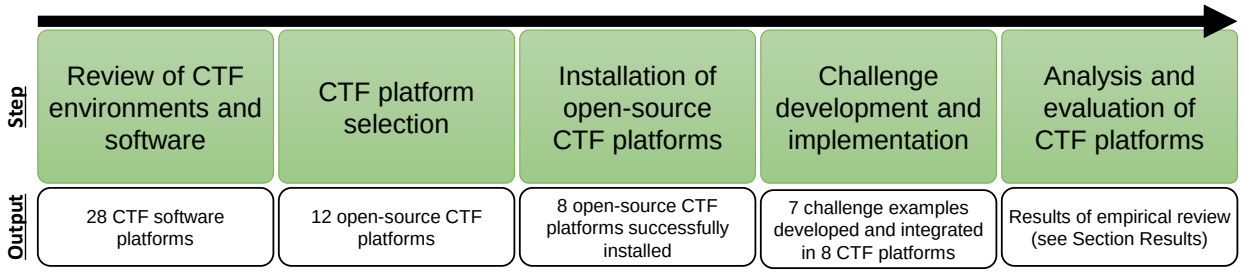

Figure 1: Methodology

publicly accessible software that has an associated open-source licence (e.g., GPL) or has no attribution. Typically the source code was publicly accessible via repositories e.g., in github[5]. On the other hand, we discovered also "proprietary" or "closed-source" software, i.e. were the owner of the source code maintains exclusive control over it and the code is not publicly accessible. We were not able to test any proprietary software in this article. From 12 open-source CTF platforms (see Table A.11 in the appendix), four could not be successfully installed as the project (source code) has not been maintained for a significant period of time (iCTF[36], OCCP[37]) or did not have any or comprehensive enough documentation (NightShade[38], OpenCTF[39]). Hence, these 4 closed-source software platforms were not included in the analysis, resulting in a set of 12 open-source CTF platforms: *PicoCTF*, *FacebookCTF*, *HackTheArch*, *WrathCTF*, *Pedagogic-CTF*, *RootTheBox*, *CTFd* and *Mellivora* (see Figure 2).

3.3. Installation of Open-Source CTF Platforms

After the platform selection, the CTF platforms were installed and assessed. Installing means successfully instantiating (setting up and starting) a CTF platform on a local machine by following the steps and instructions provided by the manual or programmers (code contributors). From the 12 selected platforms, could be 8 have been installed and tested. Only 8 platforms could successfully installed because the technology had been held up-to-date and other installation steps and support were provided on a website or in a forum. Another important aspect is that when needed, the contributors of the software offered support and help through the exchange platform (GitHub) and mail. Four platforms, namely: *iCTF*, *NightShade*, *OCCP* and *OpenCTF*, did not meet these criteria and could therefore not be included in the analysis.

The next step was to develop challenges to fully assess the platforms.

3.4. Challenge Development and Implementation

In order to investigate the CTF platforms, 16 challenge examples have been developed based on research and experience in cyber security and CTFs. These challenges have been implemented to determine which challenge types (quiz, jeopardy and king of the hill) are supported by the platforms, and to examine the provided challenge configuration options. The preparation and description of examples are outlined in Appendix B. In particular, we describe 2 *king of the hill* (K), 4 *jeopardy* (J) and 1 *quiz* (Q) examples. We have created 9 more quizzes but since the format of this type is rather straightforward, we have chosen only one representative example. A brief overview of the examples and their content is given in Table 3.

The examples were developed for participants with beginner and intermediate level in cyber security education. The findings regarding the supported challenge types and the different configuration options are covered in Section 4.3.2.

It is important to note that the challenge example set are examples related to cyber security, network and application security. However, there are numerous other examples of challenges available for beginners, intermediate or expert levels (e.g., [40, 41, 42, 43, 44]).

3.5. Analysis and Evaluation of CTF Platforms

The final step was the analysis and evaluation of the 8 CTF platforms. We conducted a thorough analysis of the supported features and technology. In addition, we utilized and integrated the

[5] `https://github.com`

Table 3: Challenge example set overview

Ex.	Type	Description	Goal
1.1	K	WordPress content injection	The goal is to edit the content of a vulnerable website and insert the team identification in the content.
1.2	K	WordPress plug-in exploitation	This foresees the exploitation of a vulnerable plug-in in order to execute PHP code and gain access to the site's content directory.
2	K	Replibit privilege escalation	The target is a backup manager by Replibit that is vulnerable to code execution and privilege escalation. The participant is prompted to exploit this vulnerability to gain root privileges and find the flag on the target machine.
3	J	Cryptography with Caesar and base64 encoding	The goal is to decode a string that was encrypted once with base64 encoding and then with a 13-Rot or Caesar cipher.
4	J	Cryptography with Vigenere and binary encoding	With the original string and the encryption given, this task challenges the participant to modify and decode the encrypted string to get to the flag which is the encryption key.
5	J	Steganography with a PNG photo	A string is hidden in the hexadecimal data representation of the picture and is to be found by the participant by analyzing the picture with a text editor.
6	J	Steganography with a GIMP-generated picture	This example focuses on decoding the hexadecimal color codes of three colored stripes which results in three parts of the flag string.
7	Q	Quiz	A simple question that requires the participant to name a specific RFC (Request for Comments).

J: Jeopardy, K: King of the Hill, Q: Quiz

developed challenges (see Appendix B) in the CTF platform to assess the compatibility as well as challenge configuration options. In particular, the CTF platforms were evaluated by

- Technology and installation prerequisites (see Section 4.1)
- Architectural overview (see Section 4.2)
- Supported features (see Section 4.3.1)
- Supported challenge types (see Section 4.3.2)
- Supported CTF game and challenge configuration options (see Section 4.3.3)

The results of the evaluation are described in Section 4 and the main findings are outlined in the discussion in Section 5.

4. Results

This section describes the results of the survey and is divided into several subsections. First, an overview of the examined 8 open-source platforms, their technology and general characteristics is given in Section 4.1. Second, an architectural overview of the platforms and supported activities are outlined in Section 4.2. Third, results of the individual analysis of the 8 CTF platforms, including supported features, challenge types and challenge configuration options, are described in Section 4.3.

4.1. Overview of Selected Open-Source CTF Platforms

This section introduces the 8 investigated open-source platforms: CTFd[45, 34], FaceboookCTF[17], HackTheArch[46], Mellivora[47], Pedagogic-CTF[48], PicoCTF[49], RootTheBox[50] and WrathCTF[51]. Table 4 describes for each platform the platform originator, year of the most recent edit and the license which their code has been published under. Screenshots of the 8 selected CTF platforms can be seen in Figure 2.

Most platforms can be installed upon an arbitrary operating system (OS), but some of them are recommended only for certain Linux distributions (see Table 5). As stated by the developers, this limitation arises mostly because the respective platforms have been developed on or for a specific OS, but this does not exclude the possibility of running successfully on other systems.

Almost all platforms require or are recommended to be installed inside a software container, namely *Vagrant*[6] or *Docker*[7] (see Table 5). This ensures modularity and independence from the underlying system, but also a smoother and easier installation process. Most platforms use `Python` as implementation language, particularly in combination with `HTML` and `Javascript`.

4.2. Architectural Overview

For a better understanding of the following content, it is important to give an design and architecture of typical CTF environments. In our investigation, we found a similar architectural abstraction layers of the CTF platforms as depicted in Figure 3. In the following, we will describe layers. The first layer is hardware that the software runs on. On the next layer, there is typically an operating system. While some platforms have been developed for specific OS, others can run on any OS (see Section 4.1). On top of an OS, there are two additional providers required in almost all selected CTF platforms: software containers and virtualization providers (cf. Figure 3). Software containers are used to isolate software and make it independent from the rest of the system, thus they *contain* the software. Popular examples are for instance *Docker* and *Vagrant*. Virtualization providers enable virtual instances of another operating system to run upon a host operating system. Well known examples of virtualization providers are *VirtualBox*[8] and *VMware*[9]. Either onto the OS or with the help of a software container (and a virtualization provider, if required), the CTF platform is installed. The platform is used to manage and provide CTF challenges. In some CTFs, CTF challenges may be located outside of the platforms. This case is however not displayed in the figure. Furthermore, different delivery models (see Section 2.2) such as in the cloud may have different abstraction layers.

4.2.1. Supported Activities

In order to investigate the supported features of the platforms further, we need to introduce the most common activities first. The main goal of this section is describe typical activities conducted with

[6] `https://www.vagrantup.com/`
[7] `https://www.docker.com/`
[8] `https://www.virtualbox.org/`
[9] `https://www.vmware.com/`

Table 4: List of selected open-source platforms

Acronym	Name	Originator	License
CTFd [45]	CTFd	Kevin Chung	Apache License 2.0
FBCTF [17]	Facebook CTF	Facebook Inc.	Creative Commons Attribution 4.0
HackTheArch [46]	HackTheArch	Military Cyber Professionals Association (MCPA)	MIT License
Mellivora [47]	Mellivora	Nakiami	GNU General Public License v3.0
Pedagogic CTF [48]	Pedagogic CTF	Hugo Delval	N/A
PicoCTF [49]	PicoCTF	Carnegie Mellon University	MIT License
RootTheBox [50]	RootTheBox	Joe D. (Alias: Moloch)	Apache License 2.0
Wrath CTF [51]	"What? Really? Another Tiny Homebrewed CTF Framework?"	Nick Frost	MIT License

Figure 2: Screenshots of 8 open-source CTF platforms

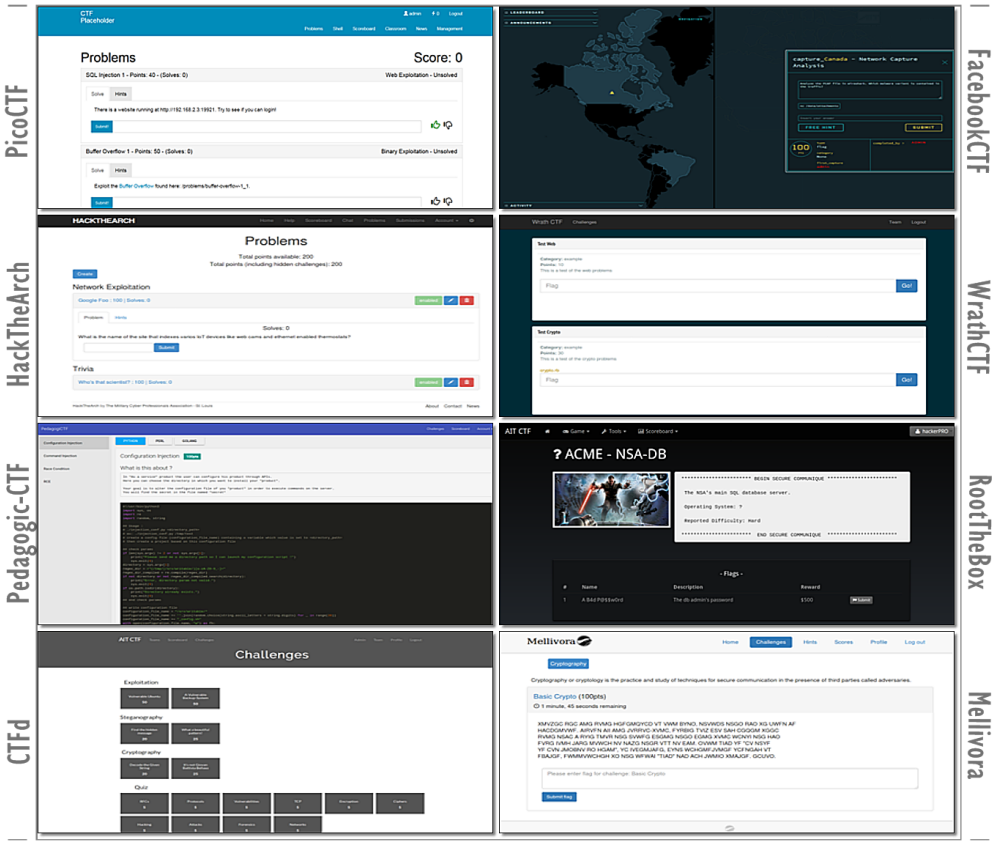

Table 5: Results: System requirements and development languages

CTF Software	OS	Installation	Language
CTFd	Any	Vagrant, Docker	PHP
FBCTF	Any	Vagrant, Docker	Python
HackTheArch	Linux	Docker	Ruby on Rails
Mellivora	Ubuntu 16.04	EC2 instance	PHP
Pedagogic-CTF	Any	Docker	Python
PicoCTF	Any	Vagrant	Python
RootTheBox	Linux, BSD, OSX	Local	Python
WrathCTF	Any	Vagrant, Docker	Python

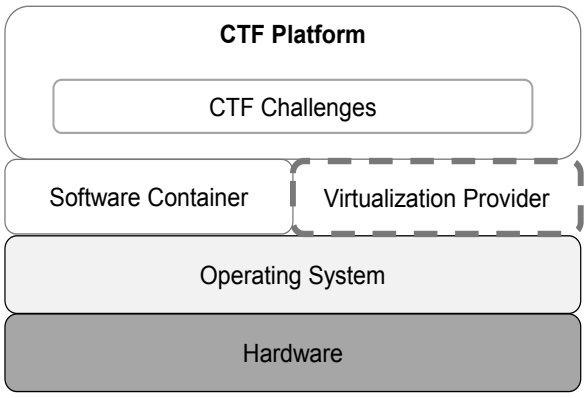

Figure 3: Architectural abstraction layers. Please note that we found that a virtualization provider is necessary when using Vagrant as the software container, therefore depicted with a dashed outline.

the platforms. Based on the timing of activities for a CTF event, we have outlined the the activities into three main categories: pre-CTF activities, in-CTF activities and post-CTF activities (see Figure 4).

Pre-CTF activities include actions that prepare the event and setup the platform and challenges. These activities include the registration of participants, team configuration, CTF event setup (e.g., interface appearance, scheduling, scoring schema), and announcements. During an active event, **in-CTF activities** are carried out and include: solving challenges, scoring (i.e. assigning points for correct answers), announcements on realtime changes in the event (e.g., when a new challenge is added, when a participant solves a challenge, or when the CTF organization wants to communicate important messages), and system monitoring (i.e. functioning of the challenges and usability of the platform) and event monitoring (e.g., keeping scores of participants and generating a scoreboard, calculating statistical data, monitoring behavior of participants to ensure fairness). **Post-CTF activities** are typically performed after the event. These activities can include backing up relevant data, event analysis and forensics which means investigating the behavior and outcomes during the event retrospectively. Lastly, communicating the results via announcements is a typical post-event activity.

4.3. Individual Analysis of Selected CTF Platforms

This section describes the detailed results of the 8 selected open-source CTF platforms by their

- supported features (cf. Section 4.3.1)
- supported CTF challenge types (cf. Section 4.3.2)
- supported CTF game and challenge configuration options (cf. Section 4.3.3)

4.3.1. Supported Features

In this article, we refer to features as units of functionality or potential configuration options in software. These features allow for realization of activities described in Section 4.2.1. Tables 6 and 7 outline the results on the supported features per platform. In the following, we will describe the results by each feature.

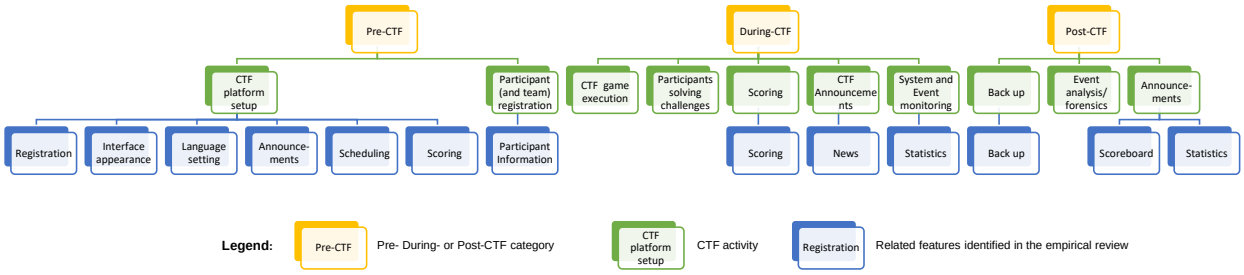

Figure 4: Common activities and related features utilized in a CTF environment

Table 6: Results: Supported features per platform (1)

Features	**CTFd**	**FBCTF**	**HackTheArch**	**Mellivora**
Interface appearance	Add pages, add CSS	-	-	Add pages, add menu items
Backup/ Restore	Both `.zip` format • Challenges • Teams • Solves, Wrong Keys, Unlocks • Configuration	Both `.json` format • Challenges • Teams • Logos • Attachments • Categories • Database	-	-
Language	EN	Multi-language	EN	EN
News	Email	Announcements on the gameboard	Chat	• News • Email
Registration	Off (not possible at all) / Open	Off (admin can register teams manually) / Open / Tokenized	Off (not possible at all) / Open	• Open • Whitelist • Blacklist
Scoring	Points	• Points • Bonus • Bonus dec.	Points	Points
Statistics	Numerical, graphical	-	Numerical, graphical	Numerical
Participant information	• Team name • Email • Password	• Team name • Player names • Player Emails • Password • Nr. of players • Logo	• Player username • Password • Team name • Passphrase	Team name • Email • Password • Country
Scheduling	Schedule	• Schedule • On click	Schedule	• per Challenge • per Category

Key:
• ... options that can be set (active) at the same time
/ ... exclusive or - only one of the options can be activated at once

Table 7: Results: Supported features per platform (2)

Features	**Pedagogic-CTF**	**PicoCTF**	**RootTheBox**	**WrathCTF**
Interface appearance	-	-	Change theme	-
Backup/ Restore	-	Export only `.csv` format Class Data	Both `.xml` format Game Data	-
Language	EN	EN	EN	EN
News	-	• News • Email	Notifications	-
Registration	Open	Open	Off (admin can register teams manually) / Open / Tokenized	Open
Scoring	Points	Points	• Points • Bot reward	Points
Statistics	-	Numerical, graphical	Numerical, graphical	-
Participant information	• Player username • Email • Password	• Player username • Password • Name • Email • Affiliation • Status	• Player username • Password • Team motto • Team name	• Team name • Email • Password
Scheduling	-	Schedule	On click	-

Key:
• ... options that can be set (active) at the same time
/ ... exclusive or - only one of the options can be activated at once

Interface appearance. Some platforms enable options for changing the appearance of the interface, i.e. the platform itself while it is running - this functionality is accordingly annotated as *interface appearance* and includes, for example, adding (HTML) pages, defining different display options of pages with CSS (Cascading Style Sheets) and adding menu items. Three of the examined platforms support this feature. For example, CTFd and Mellivora support including HTML pages, CTFd allows to design pages and to apply individual CSS. Mellivora also supports adding Menu items which can be setup to refer to a HTML page inside the platform. RootTheBox supports changing the entire appearance of the platform by allowing you to choose from a list of themes (styles). When adapting to or including a target user group which suffers from a certain color blindness, this can be an important aspect.

Backup/restore. A very important functionality which enables prevention of data loss and restoring existing settings is the backup/recovery functionality. In most CTF platforms, the backup is realized by exporting (and later restoring) relevant data. Four of the platforms support some form of backup and/or recovery, and all of them use individual file formats: `CSV`, `JSON`, `XML` and `ZIP`. In CTFd and FacebookCTF (FBCTF), you can choose which data to export/import (e.g., challenges, team data, game configuration), while RootTheBox you all game data can be exported/imported. PicoCTF only supports exporting data. This result shows that there is currently no standard interface or format for the export and import of CTF data.

Language settings. Language settings are a key factor when it comes to usability and usage of the platform. Out of the eight examined platforms, only FBCTF supports multiple languages, the rest support only English. The importance of this issue is the fact that a user who is not familiar with the English language may not comprehend the content of the CTF platforms easily. This might be relevant to younger CTF participants whos maternal language is not English.

CTF announcements and news. Communication is important particularly during CTF events. Some platforms allow predefined forms of communication between the CTF organizers and CTF participants, often called *news*, and can be realized by mail, posts on the platforms homepage or in a chat. News can be used to inform participants about other participants' activities, new events or other relevant information from the CTF organizer. Six out of 8 platforms support some kind of communication. Mellivora and PicoCTF support news in form of posts on the homepage and emails. HackTheArch is the only platform which supports instant, realtime messaging using a chat.

Registration of participants. Another central feature of CTF platforms is the registration of users for CTF events. There are several ways CTF platforms handle registrations:

- Open: users (players) can register themselves without restrictions. This option is supported by all 8 CTF platforms. Additionally, four of the platforms allow turning the registration feature off.
- Token-based: allows the administrators to generate tokens and provide them to the respective teams for the registration process. FBCTF and RootTheBox support token-based registrations.
- White-/blacklisting: collections to which user mails can be added, to either explicitly allow or prohibit their registration on the platform. Only Mellivora supports black- and whitelisting, both of which can be set up simultaneously in an open registration.

Scoring. There are certain scoring schemes, i.e. a reward system for CTFs such as for correct submissions to challenges. All examined platforms support the assignment of points. FBCTF additionally supports bonus points with a decrement-rule. In particular, bonus points are assigned to the participant who captures the flag first, the next one (i.e. the second participant) who captures the same flag gets decreased bonus points (i.e., bonus points minus the decrement points). RootTheBox additionally supports bot rewards for planting a bot (e.g., a Python script) running on a target system. Then, these points are rewarded periodically to teams as long as their bot is running on the target machine.

Statistics. Statistics may help assessing the difficulty level of challenges or the level and status of the CTF. Statistical figures such as the number of solved challenges or the ratio of solved challenges per categories are numerically or graphically

displayed in some platforms. Five platforms (i.e., CTFd, HackTheArch, Mellivora, PicoCTF, RootTheBox) provide statistical data in numerical and graphical format (see Tables 6 and 7).

Participant information. Participant information (ie. attributes such as user name, password, email, etc.) may be stored by CTF platforms when they register. The results showed that all platforms store some typical attributes such as email and/or user name and password. Further attributes are e.g., number of players (in a team), logo (avatar) of the team, passphrase (for becoming a member of a team), country or affiliation. The utilized data attributes depend on the target user group, usage and context of the platform. For example, PicoCTF, designed for educational purposes, uses the concept of roles in classrooms, a user is either an instructor or a student (other roles can be defined in the backend). Students may solve challenges belonging to a class.

Scheduling. Another feature is the scheduling of CTF events. The CTF platforms support the following options:

- setting a schedule by specifying start/end date and time of the event (CTFd, FBCTF, HackTheArch, PicoCTF),
- starting/ending an event manually per click (RootTheBox), or
- setting up a “schedule" individually for each challenge and category (Mellivora)

This analysis shows that the CTF platforms have common features and support main activities in individual instantiations. Please refer to Section 5.1 for a summary of the main findings.

4.3.2. Supported Challenge Types

After reviewing the general features of the CTF platforms, we aim to take a closer look at the challenge configuration in CTF platforms (cmp. Section 4.2.1). Therefore, we evaluate the platforms in terms of challenge types (cf. Section 2.1).

The results of the supported CTF challenge types is outlined in Table 8. It can be seen from the table that the quiz and jeopardy challenges in this review could be integrated and tested in all platforms. One reason for this might be that quizzes and jeopardy challenges require only a few input fields such as text, files or others to be implemented. These challenges are typical for CTF events and are supported by all platforms.

Table 8: Results: Supported types of CTF challenges per platform

	Q	J	K
CTFd	☒	☒	☐
FBCTF	☒	☒	☒
HackTheArch	☒	☒	☐
Mellivora	☒	☒	☐
Pedagogic-CTF	☒	☒	☐
PicoCTF	☒	☒	☐
RootTheBox	☒	☒	☒
WrathCTF	☒	☒	☐

Legend:
☒ ... supported
☐ ... not supported
Q: Quiz, J: Jeopardy, K: King of the Hill

Furthermore, our analysis showed that king of the hill CTF challenges are supported only by FBCTF and RootTheBox. One reason for this might be that this type requires more preparation due to the concept: a remote, vulnerable machine is considered a target, and the goal of participants is to capture this machine. Therein lies the tricky part: how does a team prove that they have successfully brought the system under their control? This proof of capture and interconnection between the CTF platform and the targeted systems requires various components e.g., monitoring. Both FBCTF and RootTheBox support such mechanisms. FBCTF provides a default scoring agent (which can be overwritten or replaced by a custom one) which reads out of a specific file on the target system. The teams prove that they have captured the system by writing their officially known team name in this file, and the scoring agent assigns points to the team whose name is written in the file at the time. RootTheBox utilizes a type of *botnet* system. The platform provides a bot and a bot monitor for each individual participant. This bot is added to the target system by the participant, and monitored with the bot monitor. The capturing team needs to run their bot script on the target and as long as it is running, the team will receive reward points.

Integration of Challenge Examples. In the next step, we integrated and implemented the challenge examples (cf. Appendix B) on each CTF platform. In order to fully assess the platforms, we used the examples to test the functionality and the results are outlined in Table 9. In the following, we will summarize the integration findings.

Example 1 is divided into two subtasks (see Appendix B.1). **Example 1.1** In this example, the participants are challenged to modify contents of a post on a website. In order to implement this challenge, one would need to develop a software program to *track* the changes on the target's website. During our review, only FBCTF supported the implementation of this example, as their scoring agent is able to read a specified file and check its contents for a team name. In this example, the file to be checked would simply be the content file of the website's post.

For Pedagogic-CTF and PicoCTF, there was not enough available documentation on challenge implementation and deployment, so it was not possible to carry out the implementation. However, based on the functionality review and the preinstalled examples in the platforms, we may assume that it is not very likely that Example 1.1 could be implemented on both platforms.

Furthermore, **Example 1.2** utilizes a flag and therefore can be implemented as a jeopardy challenge which is compatible with all platforms. FBCTF and RootTheBox are at the time of the investigation the only platforms to support the king of the hill challenges as described above (see Section 4.3.2). It can be seen from the Examples 3-9 in Table 9 that all platforms, except Pedagogic-CTF and PicoCTF, support quiz and jeopardy challenges and some king of the hill challenges. Based on the available functionality and supported options for designing challenges, we can infer that the Examples 1.2, 2, 3, 4, 5, 6 and 7 could be implemented in Pedagogic-CTF and PicoCTF as well. However, there was not enough documentation available to implement it.

4.3.3. CTF Game and Challenge Configuration

The CTF game and challenge configuration provided by the CTF platforms is summarized in this section. For example, game information such as instructions for the user, flags (a character string or another format), as well as reward points for successfully solving a challenge may be configured. These and other settings are identified as supported configuration options and are described below.

In this article, we use two categories to distinguish the options: if an option occurs in more than four CTF platforms (i.e. at least half of the examined platforms), it is considered a common option. If an option is supported by four or less platforms, it is considered an additional option.

Common Configuration Options We discovered that the most common configuration option in the selected open-source CTF platforms are:

- *Challenge description* explains the goals and tasks of the challenge.
- *Challenge visibility* is an on/off switch to select the visibility of the challenge to the participants.
- *Flag* is the one correct answer or solution to a challenge. There might be multiple flags correct (see Section 4.3.3).
- *Hints* are helpful comments towards the solution or clarification of the challenge.
- *Points* represent the reward for a correctly solved challenge, typically represented by a number.
- *Title* describes the heading of a challenge.
- *Upload* supports submitting files of various formats.

Table 10 displays the results of the evaluation of the common and additional options (cf. Section 4.3.3). It can be seen from the table that each CTF platform has a slightly different configuration. Figure 5 displays the frequency of options in CTF platforms.

Additional Configuration Options Furthermore, we found several additional configuration options that are not often implemented in CTF platforms:

- *Case sensitivity* allows the organizer to set whether a flag is case sensitive.
- *Dependencies between challenges* represent a *relies-on* relationship between challenges, i.e. when a challenge depends on the solution of another challenge - it becomes available when another challenge has been solved.

Table 9: Results: Implementation of challenge examples

Example	**1.1**	**1.2**	**2**	**3**	**4**	**5**	**6**	**7**
Challenge Type	K	K	K	J	J	J	J	Q
CTFd	✗	✓	✓	✓	✓	✓	✓	✓
FBCTF	✓	✓	✓	✓	✓	✓	✓	✓
HackTheArch	✗	✓	✓	✓	✓	✓	✓	✓
Mellivora	✗	✓	✓	✓	✓	✓	✓	✓
Pedagogic-CTF	✧	✦	✦	✦	✦	✦	✦	✦
PicoCTF	✧	✦	✦	✦	✦	✦	✦	✦
RootTheBox	✗	✓	✓	✓	✓	✓	✓	✓
WrathCTF	✗	✓	✓	✓	✓	✓	✓	✓

Legend:
✓ ... successfully implemented
✗ ... not supported
✦ ... not implemented, but assumed possible
✧ ... not implemented, but assumed not possible
J: Jeopardy, K: King of the Hill, Q: Quiz

Table 10: Results: Supported options for configuring CTF challenges per platform

Common Options	**CTFd**	**FBCTF**	**HackTheArch**	**Mellivora**	**Pedagogic-CTF**	**PicoCTF**	**RootTheBox**	**WrathCTF**
Challenge description	✓	✓	✓	✓	✓	✓	✓	✓
Challenge visibility	✓	✓	✓	✓	×	✓	✓	×
Flag	✓	✓	✓	✓	✓	✓	✓	✓
Hints	✓	✓	✓	✓	×	✓	✓	×
Points	✓	✓	✓	✓	✓	✓	✓	✓
Title	✓	✓	✓	✓	✓	✓	✓	✓
Upload	✓	✓	✓	✓	✓	✓	✓	✓
Upload size limited	×	✓	✓	✓	×	×	×	×
Additional Options	**CTFd**	**FBCTF**	**HackTheArch**	**Mellivora**	**Pedagogic-CTF**	**PicoCTF**	**RootTheBox**	**WrathCTF**
Case sensitivity	×	×	✓	×	×	×	×	×
Dependencies between challenges	×	×	×	✓	×	×	✓	✓
Hint penalty	✓	✓	✓	×	×	✓	✓	×
Limit number of attempts	✓	×	✓	✓	×	×	×	×
Multiple flags	✓	×	×	×	×	×	✓	×
Multiple hints	✓	×	✓	✓	×	✓	✓	×
Time frame for solving	×	×	×	✓	×	×	×	×
Waiting time between re-submissions	×	×	×	✓	×	×	×	×

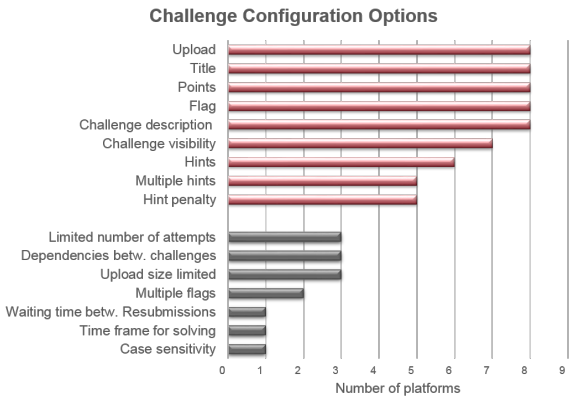

Figure 5: Frequency of challenge configuration options in CTF platforms. The common options are represented in red, and the additional options in gray.

- *Hint penalty* is the number of demanded points in return for a provided hint.
- *Limited number of attempts* means that the administrator can limit the number of times a team can submit an answer for a challenge.
- *Multiple flags* enable the implementation of multiple-choice quizzes with multiple correct answers for a challenge.
- *Multiple hints* allow the organizer to provide more than one tip for a challenge.
- *Time frame for solving* defines a certain time span for which the challenge will be available and accessible for participants.
- *Waiting time between re-submissions* specifies the minimal time which needs to pass before a team can submit to the same challenge again.

It can be seen from Table 10 that the CTF platforms differ in particularly in the additional options. Hence, it is important to CTF organizers to know which platforms offer which game configurations in order to pick the optimal tool.

5. Discussion

This section contains a discussion of the main results of this survey, its impact on the research and current practices, and limitations of the survey.

5.1. Main Findings

5.1.1. RQ1: What are typical software environments for CTF competitions?

As a conclusion from the review of the installation process and language of development (see Section 4.2), we can observe that most of the platforms can be installed upon any operating system. They mostly require a software container such as *Vagrant* or *Docker* which enables independent installation of the platform's software, regardless of the underlying system. Mellivora was the only platform which has a pre-built instance of *Amazon Elastic Compute Cloud (EC2)*. We noticed that the platforms which can be installed locally require more effort in terms of configuration such as manual installation of dependencies and requirements. The language of development was mainly Phython. Specifically, 5 platforms were implemented in Python, 2 in PHP (CTFd, Mellivora) and one with the framework Ruby on Rails (HackTheArch).

5.1.2. RQ2: Which design features and functionality does CTF software have?

We found 9 features that CTF software supports (see Section 4.3.1): (1) interface appearance, (2) backup functionality, (3) language, (4) CTF announcements and news, (5) registration of participants, (6) scoring, (7) statistics, (8) participant information, and (9) scheduling. In the following, we will review selected features:

Backup functionality is supported by four out of eight platforms and the data is stored in four different formats - ZIP, JSON, CSV, and XML. This makes it impossible to migrate data between platforms, since different formats indicate incompatibility in which case the data would need to be parsed. Also, since the fields (i.e. attributes) of challenges differ as well, it can be assumed that this would also cause some problems regarding migration.

Another important aspect is the support for **CTF announcements and news** between the admins (or the organizers) and the participants, which is supported in six out of eight platforms and it is realized with: email, notifications/posts and chats. Six out of eight platforms support this feature. Again, there is no universal, standardized way of communication. Users have to adapt to the platform's features when participating in a CTF.

Furthermore, seven out of the eight platforms only support English. Only FBCTF platform supports **multiple languages** which makes it comprehensible and usable world-wide.

Planning a CTF event in terms of setting up the start date and time of the CTF is enabled in platforms which support **scheduling**. Scheduling a CTF event for automated starting is supported by four platforms: CTFd, FBCTF, HackTheArch and PicoCTF. Manual starts are supported by FBCTF and RootTheBox. Mellivora supports scheduling on a challenge- and category level, but not scheduling for an entire CTF event. This means that by setting certain (start- and end-)date and time for a specific challenge, this challenge is available for solving in this particular time frame. When setting this kind of schedule for a category, the schedule will refer to every challenge in this category.

5.1.3. RQ3: Which challenge types are supported in CTF software?

Our analysis showed that quiz and jeopardy CTF challenges are supported by all platforms. This may be a result due to the minimal requirements needed to support these challenge types. Quizzes are typically question-answer pairs. One can consider quizzes as simplified, minimalistic jeopardy tasks, since they are independent and simple, but contain a lot less information and are easier to solve (cmp. Section 2.1). In our empirical analysis, the king of the hill CTF challenges are supported by Facebook CTF and RootTheBox. Both provide a monitoring component that should be able to track targeted systems and acknowledge if targeted systems were successfully taken over.

5.2. RQ4: How can CTF games be configured in CTF software?

With the empirical review, we were able to identify 16 game configuration options which we categorized 9 as common challenge configurations and 7 as additional options (see Section 4.3.3). We found that the most commonly available challenge configuration options are description, flag, hints, points, title of the challenge and a file upload as a sort of attachment to a challenge. They could be identified in all eight selected CTF platforms.

Additional options occurred in less than four examined platforms and some of them are specific to a particular platform such as explicitly setting case sensitivity in HackTheArch, or setting up a certain time frame for solving challenges (this also refers to the feature of setting a schedule for a certain challenge or category of challenges, described in the previous section) and setting an amount of waiting time between two sequential submissions of the same participant in Mellivora. Furthermore, only three out of the eight platforms support defining dependencies between challenges. *Dependency* essentially means that solving one challenge is the prerequisite for accessing another. This is an important aspect because with dependencies one can create different levels of challenges according to their complexity and difficulty, therefore providing better modularity and structure of the challenges as well.

5.3. Impact on research and practice

This survey includes information on the current practices regarding open-source CTF platforms, i.e. environments for organizing CTF training and competitions, and provides a comparison of platforms' functions and features, as well as CTF challenge configuration options. Since up until now no such survey has been conducted, this article can benefit other researchers in this particular field, considering cyber security exercises and CTF events used to host these exercises. CTF competitions and training are a very popular form of education and practice in the context of cyber security. The open-source frameworks that we have found can be utilized by interested organizers who want to create and host CTF events themselves, they can use information provided in this article to chose an appropriate tool for their needs. Furthermore, this article can be a good source for other interested individuals who want to take up cyber security and would like to know more about CTF exercises and events offering them. Programmers or contributors of open-source software can also benefit from the comparison we have provided in this article, in terms of improving existing or developing new software, or simply collecting ideas for future work.

5.4. Limitations of the Survey

CTF platforms that are closed source or commercial, that is, whose code is not publicly available, were not examined further in this survey, but just documented as currently existing at the time the research was conducted. This article encompasses only the CTF platforms that we have found, there might be more open-source frameworks that were not available online. Furthermore, out of the twelve discovered open-source platforms, our survey is limited to the eight platforms with sufficient documentation, that were maintained regularly and do not

contain significant bugs or issues. The survey does not include a comparison between open-source and closed-source CTF platforms as we could asses the source only with open-source platforms, nor does it examine any closed-source platforms in a particular manner (closed-source platforms were out of scope, see Section 3). The functionalities and features that have been analyzed and tested are limited to the ones that we have identified and considered significant. However, there might be others that we have not taken into account in the scope of this survey. Moreover, the challenges we have developed for the purpose of testing the configuration possibilities in the platforms are simple examples of the given cyber-security problems and there are certainly other, more complex variations and forms of these challenges that could be used for demonstration (see, for example: `https://ctf101.org/`). Nevertheless, this survey can be utilized as a comprehensive information resource for creating and/or using new technologies with better and more functional features and architecture.

6. Conclusion

This article presents a survey and comparison of open-source capture-the-flag frameworks. In particular, out of originally 28, we assessed 8 platforms where the source was accessible. The platforms that did not have enough documentation or support could not be included in a detailed analysis and, consequentially, in the comparison. The platforms we *did* get to explore further were namely (1) *PicoCTF*, (2) *FacebookCTF*, (3) *HackTheArch*, (4) *WrathCTF*, (5) *Pedagogic-CTF*, (6) *RootTheBox*, (7) *CTFd* and (8) *Mellivora*. We found that the platforms have different approaches regarding the extent of supported general features (cf. Section 4.3.1) or CTF game configurations (cf. Section 4.3.3). All of the platforms provide basic functions for setting up simple CTF events which include: registering new participants, providing a set of CTF challenges, communicating instructions and managing an underlying scoring system for assigning points. The differences amongst the platforms lie in the extensions of these functions, i.e. concrete realizations: what type of registration is supported, which options do you have while setting up a challenge, how do the users of the platform communicate and other concerns specific for CTFs. Providing a systematic overview and specific insights into functions and features of open-source CTF environments, this survey can benefit any individual, private or corporate, interested in this topic. However, in the end it is up to the CTF organizers to decide how to setup and design a CTF environment and the challenges.

Acknowledgements

This work was partly funded by the projects ACCSA and iDev40. ACCSA (860649) has been funded by the Austrian security research programme KIRAS of the Federal Ministry for Transport, Innovation and Technology (bmvit). The iDev40 project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 783163. The JU receives support from the European Union's Horizon 2020 research and innovation programme. It is co-funded by the consortium members, grants from Austria, Germany, Belgium, Italy, Spain and Romania.

Appendix A. List of Reviewed CTF Platforms

Table A.11 shows a listing of all platforms that were found during research. The platforms marked yellow are *open-source*, while the others are "proprietary" or "closed source". Additionally, We5ter [52] lists a set of CTF platforms.

Appendix B. Challenge Example Set

This section gives a brief overview of the representative instances of the developed challenge example set and their setup procedures. This includes 2 *king of the hill* examples, 4 *jeopardy* examples and 1 quiz example (cmp. Section 2.1). We have created 9 more quiz examples, but since the format of this type is rather straightforward (question-answer pairs containing text), we have chosen only one representative example. The examples were developed for participants with beginner and intermediate level in cyber security education.

It is important to note that the challenge example set are examples related to cyber security, network and application security. However, there are numerous other examples of challenges available for beginners, intermediate or expert levels (e.g., [40, 41, 42, 43, 44]).

Table A.11: CTF platforms, alphabetically ordered. The yellow marking identifies open-source software.

Acronym	Name	Originator	Delivery model	Year	License
CTF365 [53]	CTF365	CTF365 sp. z o.o.	Online Training	2017	-
CTFd [18]	CTFd	Kevin Chung	Hosting Service	2017	-
CTFd [45]	CTFd	Kevin Chung	Local Installation	2017	Apache License 2.0
DEF CON CTF [54]	DEF CON CTF	DEF CON Communications, Inc.	Live Competition	2017	-
EasyCTF [55]	EasyCTF	Michael Zhang	Online Competition	2017	-
FBCTF [17]	Facebook CTF	Facebook Inc.	Local Installation	2017	Creative Commons Attribution 4.0
Google CTF [56]	Google CTF	Google Inc.	Online Competition	2017	-
Hacking-Lab [57]	Hacking-Lab	Security Competence GmbH	Online Competition	2017	-
HackTheArch [58]	HackTheArch	Military Cyber Professionals Association (MCPA)	Live Competition	2017	-
HackTheArch [46]	HackTheArch	Military Cyber Professionals Association (MCPA)	Local Installation	2017	MIT License
iCTF [36]	UC Santa Barbara International Capture The Flag	The Security Lab at UC Santa Barbara	Local Installation	2015	GNU General Public License v2.0
Mellivora [47]	Mellivora	N/A	Local Installation	2017	GNU General Public License v3.0
MITLL CTF [59]	MIT Lincoln Laboratory CTF	MIT Lincoln Laboratory	Live Competition	2013	-
NightShade [38]	NightShade	N/A	Local Installation	2017	N/A
OCCP [37]	Open Cyber Challenge Platform	The University of Rode Island DFCSC (Digital Forensics and Cyber Security Center)	Local Installation	2015	GNU General Public License v3.0
OpenCTF [39]	OpenCTF	Michael Zhang	Local Installation	2017	MIT License
Pedagogic CTF [48]	Pedagogic CTF	Hugo Delval	Local Installation	2017	N/A
PicoCTF [60]	PicoCTF	Carnegie Mellon University	Online Competition	2017	-
PicoCTF [49]	PicoCTF	Carnegie Mellon University	Local Installation	2017	MIT License
Plaid CTF [61]	Plaid CTF	Plaid Parliament of Pwning	Online Competition	2017	-
polictf [62]	polictf	Tower of Hanoi	Live and Online Competition	2017	-
RootTheBox [63]	RootTheBox	Joe D. (Alias: Moloch)	Live Competition	2016	-
RootTheBox [50]	RootTheBox	Joe D. (Alias: Moloch)	Local Installation	2017	Apache License 2.0
RuCTFE [64]	RuCTFE	HackerDom Team	Online Competition	2017	-
SECCON [65]	SECurity CONtest	Japan Network Security Association	Live and Online Competition	2017	-
ShallWePlayAGame? [66]	ShallWePlayAGame?	Giovanni Vigna at UCSB and Adam Doupé at ASU	Hosting Service	2017	-
The Hacking Dojo [67]	The Hacking Dojo	Thomas Wilhelm	Online Training	2017	-
Wrath CTF [51]	"What? Really? Another Tiny Homebrewed CTF Framework?"	Nick Frost	Local Installation	2017	MIT License

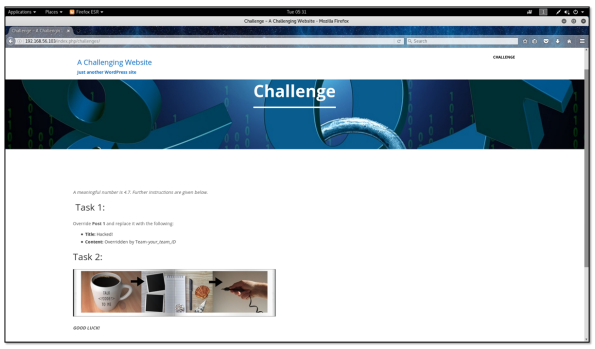

Figure B.6: Example 1: WP content injection and plug-in exploitation (Screenshot)

Appendix B.1. King of the Hill

A *king of the hill* challenge is characterized by a vulnerable system that is to be captured (see Section 2.1).

Example 1: WordPress Content Injection and Plug-in Exploitation. The chosen weakness for the target system is WordPress version 4.7, which is vulnerable (among other concerns) to *User Information Disclosure via REST API*, described in CVE-2017-5487 [68], which enables content injection (see more details on WordPress (WP) 4.7 vulnerabilities in [69]). This vulnerability enables viewing, editing or deleting content from the affected WordPress site without required user credentials. Another interesting vulnerability arises from this in combination with a vulnerable plugin for PHP code insertion: when this vulnerable plugin is exploited, scripts or arbitrary code can be injected into the post content and executed. In this example, these two "weak spots" of WP version 4.7 were prepared for exploitation on the target system, organized as two subexamples of this challenge:

Example 1.1 focuses on overriding a specific post on a WP blog and

Example 1.2 centers on exploiting a vulnerable plug-in to access the WP installation directory.

A screenshot of the vulnerable WordPress website is shown in Figure B.6.

Preparation To prepare this challenge, we created a Ubuntu Server 16.04.3 LTS (command-line-only) VM using VirtualBox. We then configured the VM so that it is visible and accessible in the internal network. Then, we set up a MySQL database (to be used by the WordPress website) and installed the (vulnerable) WordPress version 4.7 on the Ubuntu server. Additionally, we configured another Linux VM (a desktop version with a graphical user interface) within the same internal network as the target server. We used this second VM to set up the WordPress website with Mozilla Firefox web browser, and also test the exploitation process after implementing the challenge. After logging onto the WordPress website with administrator credentials, we downloaded and activated the vulnerable "Insert PHP" plugin via the plugin section of the settings. After setting up the target server and the WordPress website as described above, we created a `flag.txt` file within the WordPress installation folder, containing the base64-encoded flag string.

Example 2: Replibit Privilege Escalation. The chosen weak entity is the Replibit Backup Manager (Version: 2017.08.03), which is vulnerable to code execution and privilege escalation, as described in CVE-2017-13707 [70] (more information on the vulnerability and the exploitation process can be found at [71]). This vulnerable application allows a user to gain root privileges via the `vi` editor. This challenge was prepared for intermediate participants, because it contains less hints and requires more effort from the user's side. Figure B.7 shows the interface of this target machine. The concept of the challenge explained in short is as follows. The default user's directory contains a `readme.txt` file which provides initial instructions and a few hints for the hacker. The idea is that the root permissions are to be gained first, and then through these, a new password for the `root` user can be set. In the root directory (accessible by the root user), there is an executable file intentionally called `runme`. When the user runs it, they are prompted to enter a "secret code" which was implicitly given in the `readme.txt` file. If the user enters the correct code, a `flag.txt` file is generated, with a binary string which then just needs to be converted to ASCII to reveal the real flag.

Preparation To prepare the Replibit Privilege Escalation challenge, a new plain Linux Ubuntu x64 VM in VirtualBox was created. On this VM, a vulnerable Replibit Backup Manager instance (Version: 2017.08.03) was installed based on a downloaded `.iso` (virtual disk image) file. In the home directory of the Replibit Backup Manager, we placed a *README* file with initial instructions

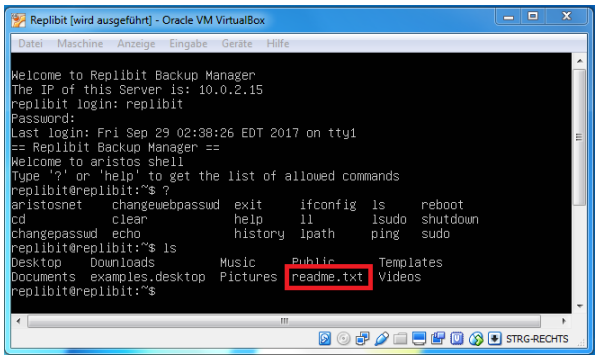

Figure B.7: Example 2: Privilege escalation (Screenshot)

```
The plaintext is:
This text will be unnecessary long elephant and exhausting to read through
until you finally arrive to the point where you realize that the flag is
actually the seventh word in this string.

The random shift number is 13

The Caesar ciphertext is:
guvf grkg jvyy or haarprffnel ybat ryrcunag naq rkunhfgvat gb ernq guebhtu
hagvy lbh svanyyl neevir gb gur cbvag jurer lbh ernyvmr gung gur synt vf
npghnyyl gur friragu jbeq va guvf fgevat.

The base64-encoded string is:
Z3V2ZiBncmtnIGp2eXkgb3IgaGFhcnByZmZuZWwgeWJhdCByeXJjdW5hZyBuYXEgcmt1bmhmZ3Zhd
CBnYiBlcm5xIApndWViaHR1IGhhZ3Z5IGxiaCBzdmFueXlsIG51ZXZpciBnYiBndXIgY2J2YWcgan
VyZXIgbGJoIGVybnl2bXIgcmd1bmcgZ3VyIHN5bnQgdmYgbnBnaG55eWwgZ3VyIGZyaXJhZ3UgamJ
lcSB2YSBndXZmIGZnZXZhdC4=

Process finished with exit code 0
```

Figure B.8: Example 3: Output of the script performing the Caesar and base64 encoding of the original string (Screenshot)

and a (camouflaged) 3-digit code. We wrote a small program in Python which prompts the user to enter the aforementioned code and, if the correct code is entered, generates a `flag.txt` file with a binary-encoded representation of the original flag string. The program was packaged as an `.exe` file to hide the source code from the participant and the user privileges are set so that only a user with root privileges can run (execute) the program.

Appendix B.2. Jeopardy

Two jeopardy examples were developed: a cryptography and steganography example. The goal of a cryptography challenge is to put the participant's encryption and decryption skills to the test by prompting them to *uncover the hidden flag by decoding a cyphered string*, while steganography challenges the user to find a hidden message in a picture file. Two challenges of each of these types were created.

Example 3: Cryptography - Caesar and Base64. In this challenge, a base64-encoded string is given. The result of decoding this string is another encoded string that is encrypted with a simple rotational cipher. Decrypting this string uncovers the flag.

Preparation This example was prepared by writing a short Python script to perform and output (see Figure B.8) the encoding of the flag string firstly with a Caeser cipher using a random shift number, and then with base64 encoding.

Example 4: Cryptography - Vigenere and Binary. This example is more complex than *Example 3*, and can therefore be considered as 'second level'. Figure B.9 shows how the description of this challenge may look like when set up for a CTF event. The user should recognize that the character string between the 8-bit binary codes are to be removed in order to get a valid binary string. Decoding the binary code to ASCII results in a ciphertext that is of same length as the original string and is ciphered with a polyalphabetic substitution, in this case: the Vigenère cipher. If one does not come to that conclusion by solely examining the string, the title of the challenge is a final clue ("It's not Giovan Battista Bellaso"). Bellaso was the one who originally described the Vigenère cipher back in 1553, but his scheme was later misattributed to Blaise de Vigenère. Since the first clue (in the original string) was that the flag is the encryption key, using the Vigenère-encoded string and any of the numerous online tools, the key of the encryption and thus the flag, can be discovered.

Preparation We prepared this challenge as follows. Using an online Vigenère cipher tool [10], we encrypted the original string "*theflagistheencryptionkey*" with the flag string "*sunrise*" as the key for the encryption. Then, we scripted a short Python program that converted the encrypted string to binary and placed randomly three character strings in between every 8-bit binary codes.

Example 5: Steganography - Photo. For this example a simple `PNG` file was used, and the 'hidden string' has been injected by editing the picture in a hex-editor, (*Hex Editor Neo*[11]). When the null spaces in the hex dump are edited, it does

[10] `https://planetcalc.com/2468/`
[11] `https://www.hhdsoftware.com/free-hex-editor`

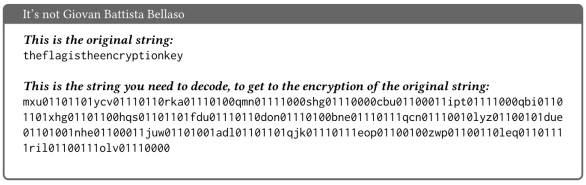

Figure B.9: Example 4: Cryptography intermediate (Screenshot)

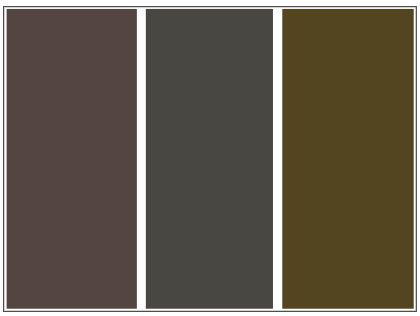

Figure B.10: Example 6: Steganography - Colors: A tricolor `PNG` created with GIMP (Screenshot)

not change the appearance of the picture. So, the idea is to simply append the hidden message to the end of the picture's hex data. This is the simplest, most trivial way to hide the string in a picture file. The hidden message contains further instructions which ultimately reveal the actual flag. Because of its simplicity, this challenge can be categorized as beginner level.

Preparation Using a hex editor, the instructions for obtaining the flag were attached at the end of the hexadecimal representation of the picture file.

Example 6: Steganography - Colors. This example focuses on the hexadecimal representations of colors and has been created with the GIMP[12] photo editor (see Figure B.10). When converted to ASCII, the color codes represent three parts of the flag-string, which can just be concatenated in the same order as the three colors.

Preparation This example was prepared as follows: we divided the flag string into three parts, converted the parts to hex codes (see Figure B.11) and then, using the GIMP software, we created

[12] `https://www.gimp.org/`

ASCII character:	T	E	C	H	G	A	T	E	!
Hexadecimal:	54	45	43	48	47	41	54	45	21

Figure B.11: Example 6: Flag string and its hex representation. The groups of three hex numbers emphasized with red boxes were used to select colors defined by the respective hex codes (Screenshot)

three colored rectangles filled with colors represented by the hex codes. Finally, the result was exported to a `PNG` file.

Appendix B.3. Quiz

The goal of this type of task is to challenge the user's knowledge, but also to provide a easy way to get points in a CTF competition. In a CTF event, challenges may additionally provide hints and/or restrictions for accepted answers. Example Appendix B.3 is an example of a quiz-pair.

Example 7: Quiz.

- **Question:** Which RFC (number) was the File Transfer Protocol (FTP) defined in?
  - **Answer:** 959

Preparation Quizzes are purely theoretical challenges prepared by composing questions with (correct) answers.

References

[1] S. Iqbal, M. L. M. Kiah, B. Dhaghighi, M. Hussain, S. Khan, M. K. Khan, K.-K. R. Choo, On cloud security attacks: A taxonomy and intrusion detection and prevention as a service, Journal of Network and Computer Applications 74 (2016) 98 – 120. `doi:https://doi.org/10.1016/j.jnca.2016.08.016`.

[2] U. Sarmah, D. Bhattacharyya, J. Kalita, A survey of detection methods for xss attacks, Journal of Network and Computer Applications 118 (2018) 113 – 143. `doi:https://doi.org/10.1016/j.jnca.2018.06.004`.

[3] R. Sahay, W. Meng, C. D. Jensen, The application of software defined networking on securing computer networks: A survey, Journal of Network and Computer Applications 131 (2019) 89 – 108. `doi:https://doi.org/10.1016/j.jnca.2019.01.019`.

[4] N. Childers, B. Boe, L. Cavallaro, L. Cavedon, M. Cova, M. Egele, G. Vigna, Organizing Large Scale Hacking Competitions, in: Detection of Intrusions and Malware, and Vulnerability Assessment, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2010, pp. 132–152. `doi:10.1007/978-3-642-14215-4_8`.
URL `https://link.springer.com/chapter/10.1007/978-3-642-14215-4_8`

[5] J. Warner, P. J. Guo, Hack.Edu: Examining How College Hackathons Are Perceived By Student Attendees and Non-Attendees, in: Proceedings of the 2017 ACM Conference on International Computing Education Research, ICER '17, ACM, New York, NY, USA, 2017, pp. 254–262. doi:10.1145/3105726.3106174.
URL http://doi.acm.org/10.1145/3105726.3106174

[6] E. Gavas, N. Memon, D. Britton, Winning Cybersecurity One Challenge at a Time, IEEE Security Privacy 10 (4) (2012) 75–79. doi:10.1109/MSP.2012.112.

[7] K. Chung, J. Cohen, Learning obstacles in the capture the flag model, in: 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14), USENIX Association, San Diego, CA, 2014, p. 7.
URL https://www.usenix.org/conference/3gse14/summit-program/presentation/chung

[8] K. Boopathi, S. Sreejith, A. Bithin, Learning Cyber Security Through Gamification, Indian Journal of Science and Technology 8 (2015) 642. doi:10.17485/ijst/2015/v8i7/67760.

[9] M. Dark, J. Mirkovic, Evaluation Theory and Practice Applied to Cybersecurity Education, IEEE Security Privacy 13 (2) (2015) 75–80. doi:10.1109/MSP.2015.27.

[10] T. Sommestad, J. Hallberg, Cyber Security Exercises and Competitions as a Platform for Cyber Security Experiments, in: Secure IT Systems, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2012, pp. 47–60. doi:10.1007/978-3-642-34210-3_4.

[11] C. Taylor, P. Arias, J. Klopchic, C. Matarazzo, E. Dube, CTF: state-of-the-art and building the next generation, in: 2017 USENIX Workshop on Advances in Security Education, ASE 2017, Vancouver, BC, Canada, August 15, 2017., 2017, p. 11.

[12] R. S. Weiss, F. A. Turbak, J. Mache, E. Nilsen, M. E. Locasto, Finding the balance between guidance and independence in cybersecurity exercises, in: 2016 USENIX Workshop on Advances in Security Education (ASE 16), Austin, TX, USA, August 9, 2016, 2016, p. 8.

[13] M. Frank, M. Leitner, T. Pahi, Design Considerations for Cyber Security Testbeds: A Case Study on a Cyber Security Testbed for Education, in: 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), IEEE, 2017, pp. 38–46. doi:10.1109/DASC-PICom-DataCom-CyberSciTec.2017.23.

[14] J. Davis, S. Margath, A Survey of Cyber Ranges and Testbeds, Tech. Rep. DSTO -GD -0771, Cyber Electronic Warfare Division, DSTO Defence Science and Technology Organisation, Edinburgh, South Australia 5111, Australia (Oct. 2013).
URL http://www.dtic.mil/dtic/tr/fulltext/u2/a594524.pdf

[15] R. Malhotra, Empirical Research in Software Engineering: Concepts, Analysis, and Applications, CRC Press, 2016.

[16] M. K. Henry, Penetration testing: protecting networks and systems, IT Governance Publishing, 2012.

[17] Facebook Inc., Fbctf, https://github.com/facebook/fbctf, accessed 14 February 2019 (2019).

[18] K. Chung, Ctfd, https://ctfd.io/, accessed 14 February 2019 (2019).

[19] Secure Iowa Conference, Capture the flag, https://secureiowaconference.com/index.php/sessions/capture-the-flag-ctf, accessed 5 February 2019 (2019).

[20] Faust CTF, Attack-defense ctf, https://2017.faustctf.net/information/faq/, accessed 5 February 2019 (2019).

[21] CTFtime team, Ctf, https://ctftime.org/ctf-wtf/, accessed 14 February 2019 (2019).

[22] Alias: Kkevsterrr, King of the hill ctf, https://github.com/Kkevsterrr/koth, accessed 5 February 2019 (2019).

[23] Alias: Irongeek, Network king of the hill ctf, https://netkoth.github.io/, accessed 14 February 2019 (2019).

[24] HackerDom Team, Phdays ctf, https://phdays.com/, accessed 14 February 2019 (2019).

[25] M. Portnoy, Virtualization Essentials, John Wiley & Sons, 2016.

[26] V. Pastor, G. Diaz, M. Castro, State-of-the-art simulation systems for information security education, training and awareness, in: IEEE EDUCON 2010 Conference, 2010, pp. 1907–1916. doi:10.1109/EDUCON.2010.5492435.

[27] S. R. Cheung, P. J. Cohen, Z. H. Lo, F. Elia, Challenge Based Learning in Cybersecurity Education, Proceedings of the Department of Computer Science, University of Massachusetts, Boston, MA, USA.

[28] M. Katsantonis, P. Fouliras, I. Mavridis, Conceptual analysis of cyber security education based on live competitions, in: 2017 IEEE Global Engineering Education Conference (EDUCON), 2017, pp. 771–779. doi:10.1109/EDUCON.2017.7942934.

[29] J. Abawajy, User preference of cyber security awareness delivery methods, Behaviour & Information Technology (2012) 237–248.

[30] L. J. Khoo, Design and Develop a Cybersecurity Education Framework Using Capture the Flag (CTF), in: W. Tan (Ed.), Design, Motivation, and Frameworks in Game-Based Learning, IGI Global, Hershey, PA, 2019, pp. 123–153. doi:10.4018/978-1-5225-6026-5.ch005.

[31] K. Bock, G. Hughey, D. Levin, King of the hill: A novel cybersecurity competition for teaching penetration testing, in: 2018 USENIX Workshop on Advances in Security Education, ASE 2018, Baltimore, MD, USA, August 13, 2018., 2018, p. 9.
URL https://www.usenix.org/conference/ase18/presentation/bock

[32] S. Wi, J. Choi, S. K. Cha, Git-based CTF: A simple and effective approach to organizing in-course attack-and-defense security competition, in: 2018 USENIX Workshop on Advances in Security Education, ASE 2018, Baltimore, MD, USA, August 13, 2018., 2018, p. 9.
URL https://www.usenix.org/conference/ase18/presentation/wi

[33] R. Raman, S. Sunny, V. Pavithran, K. Achuthan, Framework for evaluating Capture the Flag (CTF) security competitions, in: International Conference for Convergence for Technology-2014, IEEE, Pune, India, 2014, pp. 1–5. doi:10.1109/I2CT.2014.7092098.

[34] K. Chung, Live lesson: Lowering the barriers to capture the flag administration and participation, in: 2017 USENIX Workshop on Advances in Security Education (ASE 17), USENIX Association, Vancouver, BC, 2017,

p. 6.
URL https://www.usenix.org/conference/ase17/workshop-program/presentation/chung

[35] M. H. b. Noor Azam, R. Beuran, Usability Evaluation of Open Source and OnlineCapture the Flag Platforms, Tech. rep., School of Information Science, Japan Advanced Institute of Science and Technology (Jul. 2018).
URL http://hdl.handle.net/10119/15342

[36] Security Lab UC Santa Barbara, ictf, https://github.com/ucsb-seclab/ictf-framework, accessed 7 January 2019 (2019).

[37] T. U. of Rode Island DFCSC, Occp, https://opencyberchallenge.net, accessed 14 February 2019 (2019).

[38] A. Ringwood, Nightshade, https://github.com/UnrealAkama/NightShade, accessed 14 February 2019 (2019).

[39] M. Zhang, Easyctf, https://www.easyctf.com/, accessed 14 February 2019 (2019).

[40] Google, Inc., Google CTF (2018).
URL https://capturetheflag.withgoogle.com/#challenges/

[41] VulnHub, Vulnerable By Design ~ VulnHub (2018).
URL https://www.vulnhub.com/

[42] DEF CON, DEF CON Quals 2018: It's a Me (May 2018).
URL http://raywang.tech/2018/05/14/DEF-CON-Quals-2018-It-s-a-Me/index.html

[43] NCC Group, The Cryptopals Crypto Challenges (2018).
URL https://cryptopals.com/

[44] SunshineCTF, SunshineCTF (2018).
URL https://www.sunshinectf.org/challenges

[45] K. Chung, Ctfd, https://github.com/CTFd/CTFd, accessed 14 February 2019 (2019).

[46] Military Cyber Professionals Association (MCPA), Hackthearch, https://github.com/mcpa-stlouis/hack-the-arch, accessed 7 January 2019 (2019).

[47] Nakiami, Mellivora, https://github.com/Nakiami/mellivora, accessed 7 January 2019 (2019).

[48] H. Delval, Pedagogic-ctf, https://github.com/HugoDelval/pedagogic_ctf, accessed 7 January 2019 (2019).

[49] Carnegie Mellon University, Picoctf, https://github.com/picoCTF/picoCTF-platform, accessed 7 January 2019 (2019).

[50] Joe D. (Alias: Moloch), Rootthebox, https://github.com/moloch--/RootTheBox, accessed 7 January 2019 (2019).

[51] N. Frost, Wrath ctf, https://github.com/WhiteHatCP/wrath-ctf-framework, accessed 7 January 2019 (2019).

[52] Wester, A curated list of awesome security platforms,including CTF/Security Response Center/Bug Tracker and so on.: We5ter/Awesome-Platforms, accessed 17 July 2019, original-date: 2017-08-08T13:06:26Z (Jul. 2019).
URL https://github.com/We5ter/Awesome-Platforms

[53] CTF365 sp. z o.o., Ctf365, https://ctf365.com/, accessed 5 February 2019 (2019).

[54] DEF CON Communications, Inc., Def con ctf, http://ddtek.biz/, accessed 5 February 2019 (2019).

[55] Michael, Easyctf, https://github.com/easyctf/openctf, accessed 7 January 2019 (2019).

[56] Google Inc., Google ctf, https://capturetheflag.withgoogle.com/, accessed 14 February 2019 (2019).

[57] Security Competence GmbH, Hacking-lab, https://www.hacking-lab.com/index.html, accessed 14 February 2019 (2019).

[58] Military Cyber Professionals Association (MCPA), Hackthearch, http://www.hackthearch17.com/#home, accessed 7 August 2017 (2017).

[59] MIT Lincoln Laboratory, Mitll ctf, https://events.ll.mit.edu/mitllctf/, accessed 7 August 2017 (2017).

[60] Carnegie Mellon University, Picoctf, https://picoctf.com/, accessed 7 August 2017 (2017).

[61] Plaid Parliament of Pwning, Plaid ctf, http://www.plaidctf.com/, accessed 7 January 2019 (2019).

[62] Tower of Hanoi, polictf, http://www.polictf.it/, accessed 14 February 2019 (2019).

[63] Joe D. (Alias: Moloch), Rootthebox, http://root-the-box.com/, accessed 7 August 2017 (2017).

[64] HackerDom Team, Ructfe, https://ructfe.org/index, accessed 7 January 2019 (2019).

[65] Japan Network Security Association, Seccon, https://2017.seccon.jp/about/, accessed 7 January 2019 (2019).

[66] G. Vigna, Shellweplayagame?, https://shellweplayagame.org/, accessed 7 January 2019 (2019).

[67] T. Wilhelm, The Hacking Dojo Training students online since 2009!, accessed 7 January 2019 (2019).
URL http://hackingdojo.com/

[68] MITRE Corporation, Cve-2017-5487, https://www.cvedetails.com/cve-details.php?cve_id=CVE-2017-5487, accessed 14 February 2019 (2019).

[69] A. Allevato, Wordpress 4.7 vulnerabilities explained, http://www.flowpress.com/wordpress-security/wordpress-4-7-vulnerabilities-explained, accessed 5 February 2019 (2019).

[70] MITRE Corporation, Cve-2017-13707, https://www.cvedetails.com/cve/CVE-2017-13707/, accessed 28 September 2017 (2017).

[71] M. Allen, Replibit backup manager - local privilege escalation, https://github.com/Wh1t3Rh1n0/exploits/blob/master/2017-08-25%20Replibit%20Backup%20Manager/README.md, accessed 14 February 2019 (2019).