Automated Flag Detection and Participant Performance Evaluation for Pwnable CTF

Manikant Singh^(✉) , Rohit Negi^(✉) , and Sandeep K. Shukla^(✉)

C3i Center, Department of Computer Science and Engineering,
Indian Institute of Technology, Kanpur, India
{manikant,rohit,sandeeps}@cse.iitk.ac.in

Abstract. The demand for cyber security awareness, education, evaluation of learning levels of students etc., has increased in the past few years. In order to meet this rising demand, several cyber security learning and training platforms have been developed. Capture the flag (CTF) platforms and cyber ranges have become primary tools that facilitate education, training and recruitment of cyber security personnel. These tools evaluate and rank the participants on the basis of challenges solved by them. A discrete evaluation mechanism focusing only on flags solved, fails to ensure that the effort and knowledge demonstrated by the participants while solving the challenge, are factored into the scoring system. Most of these tools do not even distinguish between participants actually solving the flags vs. those who might copy a captured flag without actually working on the problem. Further, in flag only scoring systems, participants feel discouraged as they fail to score without finding the flags – despite putting in enormous time and effort. In this paper, we present our novel approach to quantify participant’s learning, efforts, and any unethical practices. We award partial scores by automatically capturing their behavior while solving the CTF problems. We also provide an accurate ranking system with automated solved challenge detection which replaces the need for manual flag submission. In our system, participants get hybrid scores based on their efforts, and organizations get a better and an effective evaluation tool.

Keywords: Cyber security · CTF · Binary exploitation · Reverse engineering

1 Introduction

Cyber security education and training play an essential role in safeguarding organizations and their users from cyber attacks. It helps in making people aware of the extent and the severity of cyber threats and encourages them to improve their security posture. Several cyber security courses and hands-on exercises have been designed to facilitate education, training & recruitment of cyber security professionals. Hands-on exercises using *capture the flag* (CTF) competitions & cyber

ranges are considered more effective as they are engaging and they acquaint the participants with core concepts in short time-frames. Consequently, experienced instructors have shifted to using hands-on learning [9] via CTFs. CTFs have been used in the cyber security domain for education as well as evaluation of learning, for over a decade [3, 5, 6, 8].

In standard CTFs, participants apply theoretical concepts to solve/exploit the challenges and capture the hidden flags which are only visible once a successful exploit of one or multiple vulnerabilities has been made.

In most CTF platforms, the participants have to submit the flags on an online portal to be awarded scores for it. If they fail to submit a flag, they get no score at all even if they spent a lot of effort unsuccessfully and almost solved it. Administering organization of the CTF use the obtained scores to evaluate, rank, select or measure learning outcomes.

Stakeholders of the CTF may differ based on their use cases. For educational use cases, the students and the instructors will be the stakeholders. In case a student obtain no score even after putting in a lot of effort, he/she might lose interest in the curriculum. On the other hand, the instructors may fail to evaluate the overall progress of their students. For a recruitment use case, the examiner, the examinees, and the hiring organization are the stakeholders. With the discrete scoring (i.e. based only on the submission of flags), hiring organizations may not get potential candidates.

Most existing tools and frameworks use discrete evaluation scheme. Most of the existing tools do not even support automated detection of capturing of flags. The first challenge is that participants in the CTF are being graded on the basis of submission of flag. Second is, they are ranked on the basis of the timestamps of the submission of flags and not on the time stamps of flag capture events. The third problem is unethical practices by the participants. It is therefore essential to have a *cyber security education and recruitment framework (CSERF)* that aims to fill these gaps by systematic evaluation of participants. An ideal CSERF must facilitate not only training exercises but also capture trainee's activities, automatically detect solved challenges, and penalize those making unethical attempts – including attack on other participants or compromising the platform itself.

In this work, we introduce a novel method to capture a participant's pursuits and quantify their efforts. This paper presents a hybrid scoring scheme where partial scores are awarded to the participants for their struggle/effort in solving the problems. Partial scores are awarded, provided that they attempt a question. Our ranking system is based on the correct sequence of events leading to solving the problem, and on penalizing detected attempts at unfair means to problem solving. Moreover, automatic detection of solved problem removes the need to create any distinct flag submission mechanism for the participants.

Rest of the paper is organized in the following manner: Sect. 2 is a brief discussion on related work. Sect. 3 describes our setup and testing environment. Sect. 4 details our evaluation scheme. Section 5 shows the results of a case study. Section 6 concludes the paper with some indications of future plans.

2 Background and Related Work

Capture the flag is one of the prominent & dominating setup used in cyber security education and evaluation. Jeopardy style CTF is the most popular type of event. As per Valdemar Švábenský et al. [11], it constitutes 86% of all the CTFs listed on ctftime.org and the challenges are further subdivided into multiple categories as mentioned in Table 1.

Table 1. CTF categories and vulnerabilities tested

Category	Vulnerabilities
Binary exploitation	Buffer/Integer overflow, format string, return oriented programming, heap exploitation, double free, use after free, etc.
Web exploitation	Broken Access Control, Broken Anti-Automation, Broken Authentication, Cross Site Scripting (XSS), Cryptographic Issues, Improper Input Validation, Injection, Insecure Deserialization, Security Misconfiguration, Security through Obscurity, Sensitive Data Exposure, Unvalidated Redirects, Vulnerable Components, XML External Entities (XXE), etc.
Network exploitation	DNS Spoofing, DNS poisoning, MITM (Man-in-the-middle), Packet Sniffing, ARP spoofing, IP spoofing, timing attack, session hijacking, etc.
Crypto	Brute force, replay, hash collision, side channel attack, etc.

Table 2. Feature comparison between different software

Features	CSERF	Pwnable.kr	CyTrOne	Hack The Box	The Juice Shop
Educational use	✓	✓	✓	✓	✓
Score board	✓	✓	✓	✓	✓
Behaviour analysis	✓	X	X	X	X
Automated flag detection	✓	X	X	X	✓
Automated scoring	✓	X	X	X	✓
Partial scoring	✓	X	X	X	X

In Table 2, we have compared several CTF tools that we used and studied by comparing their features against desired features. CTFd [4] is an open-source platform that helps to host CTF with user and flag score management capabilities. In the past few years, it has become a de-facto standard to host capture the flag tournaments via CTFd. It lets participants to submit flags for each

problem they solve and unlock new challenges as they progress. CyTrOne [1] was developed by the Cyber Range Organization and Design (CROND) at the Japan Advanced Institute of Science and Technology (JAIST). It is a cyber security training framework that simplifies the training setup process by integrating training content and training environment management. Open Cyber Challenge Platform (OCCP) is an open-source platform. It is used to educate and train high school/college students about cyber attacks by recreating already discovered attacks in a controlled environment. Participants defend/attack/investigate a network and data center with realistic attacks, but the community no longer supports it. Pwnable.kr [3] is a non-commercial war game website specialised in hosting pwnable CTF. It's similar to OverTheWire website but with some additional graphics to make it more engaging for the end-user. Another online platform is HackTheBox which allows participants to test their penetration testing skills and have a global scoreboard for teams worldwide. Multiple companies have used it for selecting new hires. All these platforms/frameworks use the conventional method of finding and submitting the flag to score.

OWASP TheJuiceShop is one of the best platforms to learn about web security. It automatically maintains a scoreboard based on the sequence of events. Once the participant accesses a particular file or performs a similar attack, it generates an event and automatically detects solved challenges. Again a discrete evaluation scheme is followed and has no partial scoring feature based on participant efforts in the right direction.

For a fine-grained evaluation, S.K. Kim et al. [7] proposed a platform to provide a fine-grained evaluation technique for pwnable CTF. They have defined four levels to evaluate student's progress: 1) Crash Check, 2) Control Flow Handling Check, 3) Mitigation Bypassing Check, and 4) Full Exploit Check. These steps are associated with the level of understanding a student might be possessing if they successfully pass a particular evaluation step. Steps are arranged in increasing order of difficulty. If a student fails at a particular level, the student is considered to have knowledge of the techniques required to cross all previous levels. Though the framework is designed to mitigate an instructor's burden, they still need to create build scripts and select the types of mitigation through a web portal. In this paper, we aim to fill the following gaps of existing CTF platforms. Most platforms provide marks solely based on flag submission, and no marks are given for efforts or evidence of knowledge and learning. Furthermore, they do not capture candidates participation behaviour data and do not apply any behavioural analytics on captured behaviour. Existing platforms do not distinguish between a flag obtained using unfair means versus flags obtained through genuine efforts.

In our work, we capture participant's activities and do behavioural analysis to generate a hybrid score. The automated multidimensional grading would enhance the capability of CTF platforms/tools and generate realistic scores and reduce the time to assess the skill set of participants. In addition, an automated flag

detection system provides more accurate rankings based on the precise time stamps of flag retrieval and reduces the possibility of using unfair means (such as copy another participant’s flags and submitting them).

Fig. 1. Platform architecture

3 Proposed Framework

This section explains our platform architecture, system setup and techniques used to extract grading parameters and grading scheme.

3.1 High Level System Architecture

As depicted in Fig. 1, high level platform architecture has two major components. The first component is the *game zone*, where all the participant machines having problem setups are deployed with auditor and logger. The second component is the *central server*, where all the logs pertaining to a participant’s activity will be processed for automated scoring. This system configuration ensures that the vulnerable machine is isolated from the Internet and cannot be used to attack other systems present on the Internet.

A single virtual machine per participant, is provisioned for the entire duration of a CTF and all the participants are given access to their separate set of vulnerable binaries to exploit. For any new challenge, we only have to add that challenge to the provisioned machines.

The Gaming Zone. It is a collection of isolated controlled virtual environments with pre-configured auditors and logging tools that record participant’s behaviour. New problems can be added to the existing virtual environment at any point in time. New virtual machines can be added to the gaming zone at the trainer/instructor’s discretion.

Centralised Server. The Centralized server is responsible for user and scoreboard management. It also consists of an Analysis engine, which is the core of the framework. Figure 2 shows different stages of analysis engine and the sequence of operations to auto-grade participants.

The Analysis engine can be considered as a collection of three sub-modules 1) Data collector 2) Analyzer 3) Auto-Grader. The first and second modules are responsible for gathering and extracting raw information. The third module utilises collected information to generate scores. The responsibilities of the first module are further divided into the following steps.

A) Update Participants List: Before retrieving the data from the server, the agent checks if there is any addition or removal of users from the server and update the local list of users. All other operations will be performed only for users present in the updated list of users.

B) Get Latest Data Logs: Once the users list is updated, the agent fetches the most recent logs generated by Osquery [10], GDB history and shell history.

C) Parse Unprocessed Logs (Analyzer): At this stage, the engine processes all the recent logs and extract the relevant information in JSON format. Later these parsed logs are utilised by the auto-grader for partial grading.

3.2 Behavioral Analysis and Data Sources

As no data set was readily available for our research purpose, we conducted a capture the flag tournament in a cyber security training course at our institute, consisting of sixty-nine participants and recorded their live data. More details of the tournament are specified in Sect. 5.

This section explains the techniques used to capture a user's behaviour and the source of each data point. The Data sources can be classified into the following four categories.

Command Line Process. We track all the processes invoked by a user either from a terminal or by a process started from the terminal. All processes invoked directly or indirectly from the terminal are labelled as command-line processes. We keep a check on the execution of all the executable files with the help of Osquery.

Fig. 2. Analysis engine

File Events. In addition to terminal driven processes, we also keep track of files accessed by a participant. Rules are set to watch for any access made to files. Keeping track of the time of access helps us determine the liveness of the participant on the server. The file path which is accessed helps us determine the binary or the file on which participant is currently working.

GDB History. GDB is an essential tool for the process of binary exploitation. Consequently, we also collect the GDB history of each user. However, by default, GDB only keeps track of commands entered by a participant, which is not sufficient for our use case. We need to know at which executable the user initiated the GDB session and how long the session lasts.

To achieve this, we have modified the GNU Debugger source code to record the executable name on which GDB is invoked and the timestamp of each command entered during that session. Using this technique, we accurately determine the debugging activities of users.

Shell History. Shell history is collected from the history of each user. This lets us determine all the activities performed by the user from the terminal.

Other data sources such as open sockets and listening ports were found irrelevant during the feature selection process. Therefore they are not discussed here.

4 Our Approach

4.1 Feature Set

In this section, we elucidate each feature extracted by our analysis engine that contributes to the final grading.

Number of Attempts: To inspect whether a participant attempted a problem or not, we look at the command line processes and the shell history. If the occurrence of a challenge’s executable file name is missing, we say that the participant didn’t attempt the question. If the occurrence is found, we keep track of its frequency. Here C is the set of all challenges, and A_{pc} denotes the number of attempts made to solve a challenge $c \in C$ by the participant $p \in P$ (Table 3). The maximum value of attempts made to solve a challenge c among all participants is given as

$$A_c^{max} = \max_{p \in P} (A_{pc}) \quad (1)$$

Table 3. Feature set with respective weight

Feature	Symbol	Weight
No. of attempts	A	W_A
Active time intervals	I	W_I
GDB active time	G	W_G
Knowledge of tools	K^t	W_{K^t}
Knowledge of system	K^s	W_{K^s}
Ethics	E	W_E

Active Time: Linux provides built-in tools to determine when the user logs in and how long the log-in session lasts. However, this gives us an overall active time but to correctly estimate the time spent by each participant on a particular problem, we propose an algorithm to generate activity periods from the logs synthetically. We have defined a parameter called buffer time (BT), which is the maximum time difference (in seconds) between two consecutive activities of the same session.


```

initialization;
 $BT$  – Buffer Time
 $I$  – Superset of active intervals, initially empty
 $P$  – Set of participants
 $C$  – Set of challenges
for  $p \in P$  do
  for  $c \in C$  do
     $I_{pc} \leftarrow 0$  /* Sum of active intervals for a challenge  $c$  of the participant  $p$  */
     $T_{pc} \leftarrow \text{Timestamps of activities for a challenge } c \text{ of the participant } p$ 
     $N \leftarrow \text{size}(T_{pc}) - 1$ ;
     $T_{pc} \leftarrow \text{sorted}(T_{pc})$ ;
     $S_{pc} \leftarrow T_{pc}^0$ ; /* Active interval start time */
     $E_{pc} \leftarrow T_{pc}^0$ ; /* Active interval end time */
     $AI_{pc} \leftarrow (E_{pc} - S_{pc})$ ; /* Current active interval */

    for  $T_{pc}^i \in \{T_{pc}^1 \dots T_{pc}^N\}$  do
      if  $T_{pc}^i - T_{pc}^{i-1} > BT$  then
        /* End previous active interval */
         $AI_{pc} \leftarrow (T_{pc}^{i-1} - S_{pc})$ ;
        /* Add active interval  $AI_{pc}$  to  $I_{pc}$  */
         $I_{pc} \leftarrow AI_{pc} + I_{pc}$ ;
        /* Update start and end time */
         $S_{pc} \leftarrow T_{pc}^i$ ;
         $E_{pc} \leftarrow T_{pc}^i$ ;
      end
      else
        /* Extend active interval  $AI_{pc}$  end time */
         $E_{pc} \leftarrow T_{pc}^i$ ;
      end
    end

    /* End last open active interval */
     $AI_{pc} \leftarrow (E_{pc} - S_{pc})$ ;
     $I_{pc} \leftarrow AI_{pc} + I_{pc}$ ;
    /* Insert sum of active interval  $I_{pc}$  into  $I$  */
     $I \leftarrow I_{pc} \cup I$ ;
  end
end

```

Algorithm 1: Active Interval Generation

Algorithm 1 shows a procedure for determining accurate active time intervals. Let us say that a participant is writing a payload script in the challenge's directory. With the help of an auditing tool, we watch for changes in a directory/file and record timestamps for each activity performed on a particular file. Using these timestamps, we generate a synthetic active time interval.

Let us say that a participant p is working on challenge c . While working on the challenge, the participant performs following actions.

- Creates a script `payload.py` at (T_{pc}^i)
- Modifies `payload.py` at (T_{pc}^{i+1})
- Deletes `payload.py` at (T_{pc}^{i+2})

If the time difference between these consecutive activities is less than the buffer time (BT) and there is no other activity in-between, we say that the participant started working on challenge c at T_{pc}^i and continued till T_{pc}^{i+2} . Therefore, they contribute to the same active interval AI_{pc} . If the time difference between two consecutive activities is greater than the buffer time (BT), we conclude the active time interval, add it to sum of existing intervals I_{pc} and begin new active interval for the current activity at time T_{pc}^i , where I_{pc} is sum of all such active intervals for participant p and challenge c . Here I_{pc} is an integer. The maximum value of cumulative interval for a challenge c among all participants is given as

$$I_c^{max} = \max_{p \in P} (I_{pc}). \quad (2)$$

This approach gives us a reasonably good estimate of a user's cumulative active time for a particular challenge.

GDB Sessions and Active Time: With our modified GDB, we have executable names on which the GDB was invoked and the timestamp of each command entered. We apply the same Algorithm 1 for synthetic timestamp sequencing on GDB activities as well, so that we have accurate active time estimates and the user is not able to manipulate active sessions. G_{pc} is a sum of all GDB active intervals for a participant p and challenge c .

The maximum value of cumulative GDB session interval for challenge c among all users is given as

$$G_c^{max} = \max_{p \in P} (G_{pc}). \quad (3)$$

Knowledge of Tools: To check if a user has used the tools required to crack the problem, we track usage of tools like `readelf`, `objdump`, `ltrace`, `strace`, `ptrace`, `hexdump`, `nasm`, `gdb` and `file`. This takes care of assessing the granularity of knowledge. The list can also be modified before starting the analysis engine. K_p^t denotes the cumulative count of selected tools used by a participant p . The maximum value of K_p^t among all participants is given as

$$K_{max}^t = \max_{p \in P} (K_p^t). \quad (4)$$

Knowledge of System: Similar to tools knowledge, we also check whether a participant understands Linux and its permission/privilege system. To take care of this parameter, we have identified few commands like `sudo`, `chmod`, `chroot`, `chgrp` and `chown`, which the participants use during an actual CTF session. However, repetitive usage of such commands indicates that the user does not understand Linux and its permission system. Commands to watch for this purpose can be added/removed at the trainer’s discretion. K_p^s denotes the cumulative count of selected commands used by participant p . The maximum value of K_p^s among all participants is given as

$$K_{max}^s = \max_{p \in P} (K_p^s). \quad (5)$$

Ethics: To identify unethical attempts, we check if participants try to access restricted inodes. Though in most pwnable/binary CTFs, participants cannot view other users’ directory content, we track such events so that the trainer gets an idea about participants trying to use unfair means. For example, commands like

```
“cat ../../home/otheruser/problem/file.txt”
“ls ../../home/otheruser/problem/”
“ls /home/otheruser/problem”
```

all are considered unethical attempts, E_p denotes the frequency of such attempts made by participant p . Participants are penalized for these activities. The maximum value of E_p among all participants is given as

$$E_{max} = \max_{p \in P} (E_p). \quad (6)$$

Solved Challenge Auto Detection: Our framework automatically detects when a participant successfully opens a “flag.txt” with elevated privileges and mark that challenge as solved for that particular participant. The system also maintains the time of the actual flag retrieval to ensure a precise ranking system. Our technique is independent of challenge type (buffer overflow, integer overflow etc.) and flag value written in flag.txt, making it easier for the tutor to have randomized flag value for each participant. Note that the auto detection mechanism for other kinds of challenges such as web security challenges will be different.

```

initialization;
/* {A, I, Kt, Ks, G, E} are features with {WA, WI, WKt, WKs, WG, WE} as
their respective weights.*/
/* Set of dedicated marks for challenges*/
CM ← {CM1, CM2 ... CMm};
F – Analysis time frame in seconds
TSD – Timestamp indicating start of the current day
SF – /* Scaling factor */
CSp ← 0; /* Cumulative score of p ∈ P */
CEp ← 0; /* Cumulative effort score of p ∈ P */
Mp ← 0; /* Marks to maintain rank of p ∈ P */
HSp ← 0; /* Hybrid score of p ∈ P */
for p ∈ P do
  for c ∈ C do
    /*normalize data */
     $\bar{I}_{pc} \leftarrow \frac{I_{pc}}{I_c^{max}}$ ;
     $\bar{G}_{pc} \leftarrow \frac{G_{pc}}{G_c^{max}}$ ;
     $\bar{A}_{pc} \leftarrow \frac{A_{pc}}{A_c^{max}}$ ;
    if c is solved then
      t – Initial timestamp when c was solved
      if t ≥ TSD then
        else
          /*Add dedicated challenge score and incentivize early solver */
           $M_p \leftarrow M_p + CM_c + SF * \left(1 - \frac{t - TS_D}{F}\right)$ ;
        end
         $M_p \leftarrow M_p + SF$ ;
      else
         $CS_p \leftarrow CS_p + \frac{W_A * \bar{A}_{pc} + W_I * \bar{I}_{pc} + W_G * \bar{G}_{pc}}{W_A + W_I + W_G}$ ;
      end
    end
    /*normalize data */
     $\bar{K}^t \leftarrow \frac{K_p^t}{K_{max}^t}$ ;
     $\bar{K}^s \leftarrow \frac{K_p^s}{K_{max}^s}$ ;
     $\bar{E} \leftarrow \frac{E_p}{E_{max}}$ ;
    /* Weighted score of knowledge and ethics */
     $CE_p \leftarrow \left( \frac{W_{K^t} * \bar{K}^t + W_{K^s} * \bar{K}^s + W_E * \bar{E}}{W_{K^t} + W_{K^s} + W_E} \right) + CS_p$ ;
  end
  CEmax ← maxp ∈ P (CEp);
for p ∈ P do
  /* Normalize cumulative effort */
   $CE_p \leftarrow \left( SF * \frac{CE_p}{CE_{max}} \right)$ ;
  /* Update hybrid score of participant p */
  HSp ← CEp + Mp;
end

```

Algorithm 2: Evaluation Algorithm

4.2 The Grading Algorithm

In the previous section, we discussed all the features which contribute to the final evaluation. Since every tutor may not find all these parameters equally relevant to them, we let them define the weights $\{W_A, W_I, W_{K^t}, W_{K^s}, W_G, W_E\}$ for each parameter. Given the weights, the auto-grader comes into action to calculate scores for all the participants. The Algorithm 2 utilizes data points $\{I_{pc}, G_{pc}, A_{pc}, K_p^t, K_p^s, E_p\}$ collected at the behavioural analysis stage to calculate these scores.

In our Algorithm 2, we loop on every participant p , and on each challenge $c \in C$. For a particular participant p , we first check if a challenge is solved or not. Our system automatically detects the first instance when the challenge was solved and imparts the dedicated challenge score CM_c and an additional incentive score to an early solver. Incentive score is calculated on the basis of the initial timestamp when a challenge was solved, as shown in the Algorithm 2. If the challenge was solved at an earlier time frame, we impart the maximum incentive marks that can be given for a time frame. The Incentive score ranges between 0 to 1 and scaled with scaling factor SF as per the configuration. Sum of CM_c and incentive score contributes to M_p , that is, the score to maintain the rank for participant p . The incentive score helps a participant maintain his/her prior rank similar to any other standard flag submission based ranking system.

If a challenge is not solved, then we calculate the partial score from the weighted sum of parameters $\{I_{pc}, G_{pc}, A_{pc}\}$ which are exclusively related to a challenge c and participant p . This weighted sum is stored in a temporary variable called cumulative score CS_p . Other parameters $\{K_p^t, K_p^s, E_p\}$ contributes to partial scores for a participant p , and the weighted score of these parameters combined with CS_p gives us the cumulative effort score CE_p for a participant p for a given analysis time frame.

Finally, hybrid score HS_p , a combination of rank maintainer score M_p and cumulative effort score CE_p , is given to the participant p . The final ranking is done using this hybrid score.

5 Case Study

Since there was no existing data set for our research and analysis, we hosted a capture the flag (CTF) tournament for a cyber security training course we run at our institute, with a total of sixty-six participants and collected logs for their activities while they were exploiting the given challenges in our controlled environment. Each participant had access to the gaming zone with pre-configured auditors and loggers. All participants actively tried to crack the given challenges, only 18.18% participants were able to solve at least one challenge, and 43.93% participants were found to have issues with tools and system.

In addition to our automated flag detection system, we gave them the facility to submit flags manually on a ranking server (CTFd) for live tracking.

Table 4 shows the ranking obtained by top 12 participants based on their flag submission and Table 5 shows the ranking obtained with our framework.

Table 4. CTFD ranking system based on flag submission

Rank	Name	Score	Last submission
1	P_{35}	3600	2021-05-08 23:51:07
2	P_{34}	3600	2021-05-20 09:26:39
3	P_5	2600	2021-05-20 02:11:09
4	P_{61}	2600	2021-05-29 17:26:43
5	P_8	2600	2021-05-29 19:14:19
6	P_{43}	2100	2021-04-27 20:36:54
7	P_{55}	1900	2021-05-16 22:47:22
8	P_{52}	1400	2021-05-30 12:26:20
9	P_{23}	1300	2021-05-28 12:56:27
10	P_{42}	800	2021-04-15 00:19:07
11	P_{12}	600	2021-04-19 13:19:36
12	P_{64}	600	2021-05-28 16:45:55

Table 5. Ranking with automated flag detection

Rank	Name	Score	Last solve
1	P_{35}	3600	2021-05-08 23:49:28
2	P_{34}	3600	2021-05-20 09:25:51
3	P_5	2600	2021-05-20 02:10:00
4	P_{61}	2600	2021-05-29 17:22:55
5	P_8	2600	2021-05-29 19:13:05
6	P_{43}	2100	2021-04-27 20:36:54
7	P_{55}	1900	2021-05-16 22:47:06
8	P_{52}	1400	2021-05-30 00:22:48
9	P_{23}	1300	2021-05-28 00:55:26
10	P_{42}	800	2021-04-15 00:15:51
11	P_{12}	600	2021-04-17 19:46:38
12	P_{64}	600	2021-05-28 16:45:25

Participants with zero submission got ranking based on hybrid scores for their efforts. This technique let us identify participants who gave significant efforts yet failed to solve. Table 7 shows top 5 participants who solved zero challenges yet got ranking from 13 to 17 based on their pursuit.

From Tables 4 and 5, we can see that there is always a delay of few minutes between flag retrieval and flag submission. Difference between flag retrieval and submission is significant in case of participant P_{12} ranked at position 11. These ranks could have been different if some other participant submitted their flag in between. Based on these results, we can conclude that ranking based on flag

Table 6. Ranking combined with hybrid scores

Rank	Name	Original score	Cumulative effort score (CE)	Hybrid score (HS = CE + M)
1	P_{35}	3600	2132.50	5732.50
2	P_{34}	3600	1359.36	4959.36
3	P_5	2600	1221.96	3821.96
4	P_8	2600	853.03	3453.03
5	P_{61}	2600	802.71	3402.71
6	P_{43}	2100	1294.26	3394.26
7	P_{55}	1900	888.81	2788.81
8	P_{52}	1400	244.15	1644.15
9	P_{23}	1300	150.30	1450.30
10	P_{42}	800	468.06	1268.06
11	P_{12}	600	476.32	1076.32
12	P_{64}	600	83.86	683.86

Table 7. Top 5 participants with zero solves

Rank	Name	Original score	Hybrid score (HS = CE + M)
13	P_7	0	161.99
14	P_4	0	109.14
15	P_{48}	0	88.50
16	P_2	0	50.48
17	P_{13}	0	48.11

submission does not fully capture and rank the participants based on the criteria we believe should be used in evaluating participants. However, our framework more accurately capture these criteria in scoring and ranking of participants. Of course, one case study is not enough to be sure, and we plan to do more extensive experiments in the future batches of trainees to get a better data support for this assertion.

Table 6 shows ranking obtained after assigning partial scores (cumulative effort scores). Here we can clearly see that partial scores works as a tie breaker and favours participants with more efforts in lesser time frame. The justification for this is that those putting more effort in solving the challenges are possibly learning more from the exercises.

6 Conclusion and Future Work

CTF tournaments are playing a crucial role in cyber-security education and training activities. We have built a system that improves the evaluation

techniques used in traditional CTF tournaments, and this paper tries to elucidate our work. Our framework's evaluation scheme motivates participants to put in more efforts and learn, through partial scores for their participation and activeness. They are not judged just on the basis of solved problems but also ethics, and their knowledge of tools and systems. From Tables 5 and 6, we can observe that with hybrid scores, we still get similar ranks as that of a discrete score based ranking system but with some significant positive discrimination for hardworking and ethical participants. From Table 7, we observe that participants who solved zero challenges also got their ranks aligned according to their efforts, which does not happen in a discrete score-based ranking system. Discrete score based systems do not provide any way to differentiate between active and non-active participants. Techniques proposed in this paper creates a favourable environment for cyber security organizations, educational institutions and participants. With solved challenge auto-detection, we have improved the ranking system based on correct sequence of events.

Although CSERF is the first of its kind to auto-grade participants, there is still a vast scope of improvement and innovation. The Framework is implemented for pwnable CTF. Nonetheless, there are numerous training challenges like networking, web, cryptography, defensive cyber security etc., where we can extend the support for hybrid scoring. Harsh et al. [2] mentioned that using server logs, one can identify participant's behaviour. Web request can be classified into attacks such as SQL Injection, Cross-Site Scripting, Path-traversal, Command Injection, Cross-site request forgery etc. We can utilize this technique to calculate the partial score of participants for web challenges also.

Also, as we gather more data over many tournaments, scoring system can be improved using machine learning models. Currently, the data being limited, we have provided algorithms based on our assumptions on how a user behaves towards solving CTF challenges.

References

1. Beuran, R., Pham, C., Tang, D., Chinen, K., Tan, Y., Shinoda, Y.: CyTrONE: an integrated cybersecurity training framework (2017)
2. Bhagwani, H., Negi, R., Dutta, A.K., Handa, A., Kumar, N., Shukla, S.K.: Automated classification of web-application attacks for intrusion detection. In: Bhasin, S., Mendelson, A., Nandi, M. (eds.) SPACE 2019. LNCS, vol. 11947, pp. 123–141. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-35869-3_10
3. Azam, M.H.B.N., Beuran, R.: Usability evaluation of open source and online capture the flag platforms (2018)
4. Chung, K.: Live lesson: lowering the barriers to capture the flag administration and participation. In: 2017 USENIX Workshop on Advances in Security Education (ASE 2017) (2017)
5. Chung, K., Cohen, J.: Learning obstacles in the capture the flag model. In: 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14) (2014)

6. Ford, V., Siraj, A., Haynes, A., Brown, E.: Capture the flag unplugged: an offline cyber competition. In: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, pp. 225–230 (2017)
7. Kim, S.-K., Jang, E.-T., Park, K.-W.: Toward a fine-grained evaluation of the Pwnable CTF. In: You, I. (ed.) WISA 2020. LNCS, vol. 12583, pp. 179–190. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-65299-9_14
8. Kucek, S., Leitner, M.: An empirical survey of functions and configurations of open-source capture the flag (CTF) environments. *J. Netw. Comput. Appl.* **151**, 102470 (2020)
9. McDaniel, I., Talvi, E., Hay, B.: Capture the flag as cyber security introduction. In: 2016 49th Hawaii International Conference on System Sciences (HICSS), pp. 5479–5486. IEEE (2016)
10. Osquery. Osquery
11. Švábenský, V., Čeleda, P., Vykopal, J., Brišáková, S.: Cybersecurity knowledge and skills taught in capture the flag challenges. *Comput. Secur.* **102**, 102154 (2021)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.