Data Collection in Cyber Exercises Through Monitoring Points: Observing, Steering, and Scoring

Tobias Pfaller[1], Florian Skopik[1], Lenhard Reuter[1] and Maria Leitner[2]
[1]*AIT Austrian Institute of Technology, Vienna, Austria*
[2]*University of Regensburg, Regensburg, Germany*

Keywords: Cyber Exercise, Cyber Exercise Scenario, Participant Observation, Activity Monitoring, Cyber Exercise Evaluation, Cyber Scenario Steering.

Abstract: Cyber security exercises are an essential means to train people and increase their skill levels in IT operations, cyber incident response, and forensic investigations. Unfortunately, carrying out high-quality exercises requires tremendous human effort in planning, deploying, executing and evaluating well-planned cyber exercise scenarios. While planning a scenario is often only a one time effort, and deployment can be highly automatized today, their repeated execution and evaluation is a resource-intensive task. Usually human experts manually observe the participants to recognize any difficulties in carrying out the exercise and to keep track of the participants' progress. This is an essential prerequisite to not only support participants during the exercise, but also to drive the scenario further through timely injects, and provide feedback after the exercise. All this manual effort makes exercises a costly activity, reduces scalability and hinders their wide adoption. We argue that with automating observations, recognizing participant progress with only little to no human effort, and even steering the delivery of customized injects, cyber exercises could be carried out much more cost-effective. In this paper, we therefore introduce the concept of monitoring points which enable the scenario-dependent collection of technical data and the calculation of behavior and progress metrics to rate participants in exercises. This is the foundational basis for steering an exercise on the one side, and evaluation on the other side. We showcase our concept and implementation in course of a demonstrator consisting of a cyber exercise comprising 14 participants and discuss its applicability.

1 INTRODUCTION

Cyber ranges and cyber exercises are important means to prepare against cyber threats by providing realistic and controlled environments for cybersecurity professionals to practice and improve their skills (Karjalainen et al., 2019). These platforms simulate real-world cyber attacks, allowing participants to respond to various scenarios, test their incident response plans, and enhance their ability to detect, mitigate, and recover from threats. Through hands-on experience in these simulated environments, individuals and teams can identify weaknesses in their cybersecurity posture, develop effective strategies for defense, and collaborate with others to strengthen their overall resilience against cyber threats.

The observation of participants is an important corner stone of every cyber exercise delivery. It allows to control injects to adapt exercises to the participants' progress and draw conclusions with respect to exercise objectives. However, current approaches for participant observation are mostly implemented as either resource-intensive manual activities, i.e., human observers are employed, or only superficially determine the status of exercises, i.e., participants need to submit reports or flags, or are interrogated with questionnaires. Furthermore, both reports and questionnaires are subjective assessments that do not allow for a detailed review of participants' activities. Manual observations allow for detailed insights, but they are a huge burden, can barely be automated, and thus scale poorly. As a consequence, the delivery of cyber exercises requires a huge amount of resources, which may be overwhelming for the organizers, but definitely increases costs of exercises tremendously.

We argue that with automating observations, recognizing participant progress with only little to no human effort, and even steering the delivery of customized injects, cyber exercises could be carried out much more cost-effective.

Meticulously observing participants in cyber exercises provides benefits in multiple dimensions:

- **Providing Feedback.** An after-action report usually summarizes how single participants or a team

Pfaller, T., Skopik, F., Reuter, L. and Leitner, M.
Data Collection in Cyber Exercises Through Monitoring Points: Observing, Steering, and Scoring.
DOI: 10.5220/0013309900003899

In *Proceedings of the 11th International Conference on Information Systems Security and Privacy (ICISSP 2025) - Volume 1*, pages 355-366
ISBN: 978-989-758-735-1; ISSN: 2184-4356

did in general solving a cyber scenario.

- **Steering.** The timed delivery of injects to drive a scenario further is usually based on the participants' progress to keep them busy on the one side but not to overburden them on the other side.
- **Scoring.** Evaluating single participants and validating their learning goals, as well as identifying opportunities for skill improvement is also based on observations during an exercise.

In this paper, we therefore introduce the concept of *monitoring points* which enable the scenario-dependent collection of technical data and the calculation of behavior and progress metrics to rate participants in exercises. This is the foundational basis for steering an exercise on the one side, and evaluating participants and exercise runs on the other side. In contrast to existing methods, our approach goes beyond analyzing individual sources of the infrastructure (e.g., the bash history) and allows holistic, targeted monitoring of a cyber exercise infrastructure.

In particular, the contributions of this paper are:

- **Concept of Monitoring Points.** We describe the concept of monitoring points within a cyber exercise infrastructure, including their various types and inter-relations.
- **Proof-of-Concept Implementation.** We showcase an example of a proof-of-concept (PoC) implementation for specific exercise tasks and highlight technical issues.
- **Evaluation and Demonstration.** We showcase the application of the PoC in course of a real-world pilot and critically discuss the use and applicability of monitoring points.

We consider our work important for everyone who realizes and delivers cyber exercises and faces scalability issues regarding the observation of potentially large groups of participants.

The remainder of the paper is organized as follows. Section 2 outlines important background and related work. Section 3 introduces the concept of monitoring points, while Sect. 4 describes the implementation of a proof-of-concept. Section 5 discusses the evaluation results of a real-world pilot and the applicability of monitoring points. Finally, Sect.6 concludes the paper.

2 BACKGROUND AND RELATED WORK

Since evaluating the performance of participants in cyber exercises is very cumbersome, many cyber exercise organisers limit their observation to superficial technical metrics, as for example whether a specific service is available or has been successfully compromised by attackers. In an e-commerce exercise (Snyder, 2006), the authors simulated customers that requested a service each minute and the exercise teams earned points at each request if the service was available. In Platoon (Li and Xie, 2016), the design of a virtual platform for cyber trainings and exercises is presented. The authors integrated a scoring engine, that calculates real-time service scores based on service availability. In a cyber exercise implemented by Vykopal et al. (Vykopal et al., 2017), penalty points where computed if services where inaccessible.

In Capture the Flag (CTF) (Kucek and Leitner, 2020) exercises, participants uncover flags – usually represented as strings – by solving predetermined tasks. These flags must then be submitted on a designated platform to earn points. Consequently, the submission of flags serves as an indicator of participants' progress and performance. Submitting flags allows for a superficial determination of when certain tasks were solved. Similarly, monitoring service availability also enables the identification of when certain tasks were completed, as for example, when a service becomes available again after an attack. Such simple evaluation methods are effective, comparable, and meaningful, making them well-suited for measuring success and scoring of participants.

However, in that context, Weiss et al. (Weiss et al., 2016) argue, that simple technical metrics like just measuring if a task was solved or not solved in cyber exercises does not give a clear indication of whether the task was fully understood nor whether the learning objectives were achieved. In order to accordingly turn cyber exercises into valuable training events, participants need more detailed feedback on their actions (Eagle, 2013). Therefore, Vykopal et al. (Vykopal et al., 2018) created a scoring timeline, to make participants aware of when and why they lost or gained points. Participants should give feedback on lost points by choosing predefined answers (e.g., "We answered this immediately." or "We had no idea what to answer."). Mirkovic et al. (Mirkovic et al., 2020) present ACSLE, a system monitoring participants progress in cyber exercises that focuses on terminal-based interactions. The authors predefined milestones and baseline commands that are necessary in order to complete certain tasks and compare participants commands to these baselines. Additionally, ACSLE is used to create metrics and predict with 80 % accuracy whether students will struggle while completing the cyber exercise or not (Vinlove et al., 2020). Similar to ACSLE, Svabensky et al. recorded

command line logs of students during a cyber exercise in order to model their progress through the cyber exercise. Based on a questionnaire for instructors, they evaluated and assessed both a trainee graph and a milestone graph in order to make implications for teaching practice. Macak et al. (Macak et al., 2022) contributed an approach utilizing a process discovery algorithm in order to discover participants processes in cyber trainings. They used the bash history of participants and extended it with "hints taken"-activities, where the participants requested predefined hints.

In this regard, the cited works primarily offer deeper insights into participants' behavior rather than merely identifying task completion. However, their logs are predominantly limited to Bash commands. While the exclusive analysis of Bash history provides a comprehensive and clear view of participants' behavior in exercises conducted solely within the Bash environment, it is often insufficient for more complex infrastructures. Cyber exercises take place on complex platforms (i.e., Cyber Ranges (Leitner et al., 2020), (Čeleda et al., 2015)), which simulate virtual infrastructures often compromising multiple networks and servers (potentially even augmented by physical components (Yamin et al., 2018)). Attacks occur somewhere within the infrastructure, and participants must possibly connect to remote servers to mitigate these incidents. In such environments, solely analyzing the Bash history proves to be a limiting factor, as crucial actions also occur outside of a participants' Bash history. Therefore, the improvement of data collection during cyber exercises is a decisive factor to allow more accurate evaluation of participant performance (Henshel et al., 2016).

Andreolini et al. (Andreolini et al., 2020) present an approach to discover and assess the performance / behavior of participants on cyber ranges. They collect data such as command history, web browsing history, GUI interactions, and network events to define events, add them to a graph and thereafter calculate metrics like speed or precision. However, the authors focus on graph development and evaluating the participants performance instead of demonstrating how to collect and process data from the exercise environment. Braghin et al. (Braghin et al., 2020) created a hierarchy of categories and sub-categories of actions participants can perform in a cyber exercise and took advantage of the approach described in (Andreolini et al., 2020) by adapting it to their purposes. They used the resulting graph to algorithmically score participants.

In addition to determining whether tasks have been completed or not and developing graphs of participants' activities, there are a plenty of technical metrics that can be calculated and utilized for evaluation and scoring purposes. Possible metrics are, for instance, the time until a certain command is executed (Labuschagne and Grobler, 2017), the mean time per action / task (Abbott et al., 2015), the number of actions per task (Abbott et al., 2015), the number of correctly identified attacks (Patriciu and Furtuna, 2009), and many more. Maennel et al. (Maennel, 2020) performed an extensive literature review and determined potentially relevant metrics and argued how they could be measured in order to reflect the learning success of participants.

While there is a substantial body of literature on participant behavior in cyber exercises, existing approaches often have significant limitations. Many approaches either focus narrowly on technical monitoring, such as analyzing exclusively the terminal history (e.g., (Mirkovic et al., 2020), (Macak et al., 2022)). Other approaches emphasize behavioral representation and comparison using graph-based methods (e.g., (Andreolini et al., 2020), (Braghin et al., 2020)), without going into detail about how the used data is collected from systems. Our approach addresses this gap by providing a comprehensive method for the targeted monitoring of complex infrastructures, enabling the collection of meaningful data and generating valuable insights from it.

3 MONITORING POINTS

In the complex delivery of cyber exercises, the challenges do not only lie in monitoring participant activities but also in deriving meaningful insights from the abundance of generated data. Our approach aims to address these challenges by leveraging the scenario-based nature of cyber exercises (Wen et al., 2021) to our advantage.

3.1 Concept of a Monitoring Point

In essence, we introduce the concept of *monitoring points*, which are strategically positioned within the exercise environment to gain enhanced insights. These points serve as focal nodes, each designed to examine specific predefined facets of a participant's system. Instead of casting a wide net across the entire cyber exercise infrastructure or limit our monitoring to single components (e.g., bash history), we advocate for targeted monitoring that focuses on components and applications relevant to the cyber exercise scenario and its learning objectives.

A monitoring point is a passive observer within the exercise environment, comprising two fundamen-

tal components: (1) *the monitoring mechanism* and (2) *the resulting logs* – a tangible artifact stemming from the monitoring mechanism. The monitoring mechanism serves as a sensory organ that perceives activities or states within a participant's system. These mechanisms are, for instance, running custom-built observer services, leveraging existing monitoring mechanisms such as Auditd (Red Hat, 2022), or observing application logs (e.g., from Apache, Nginx, MySql, Command Line, or other services). In case of using existing monitoring mechanisms, they may require custom configurations – such as configuring Auditd rules or extending the default Bash history (e.g., with timestamps).

In order to give coherence to resulting logs of multiple different monitoring points, they can be related to each other by the relationships *aggregation* or *influence*. An *aggregation* allows for combining the results of multiple monitoring points into one aggregated, more expressive result. An *influence* reflects that activities observed by one monitoring point (or multiple aggregated monitoring points) influence the state of another monitoring point. These relationships enable (1) to aggregate the results of multiple monitoring points to one central point, and (2) to relate activities of monitoring points to state changes. For instance, an aggregation of monitoring points that monitors a directory containing configuration files of a certain service is in an influencing relationship to a monitoring point that observes the status of this certain service. Therefore, our approach enables to determine not only whether the service status has changed but also what actions were undertaken by participants to cause it.

3.2 Formal Model of Monitoring Points and Their Relationships

The following subsections comprise the formal model of a monitoring point and of relationships between monitoring points.

3.2.1 Monitoring Point

Let MP_i denote a monitoring point, defined by the tuple (T_i, S_i, L_i), where

- T_i denotes the type of the monitoring point's results, which can be either "Activities" or a "State". If $T_i =$ "Activities", it signifies that the monitoring point captures activities of participants (e.g., actions on the file system). If $T_i =$ "State", it signifies that the monitoring point's results represent the state of a specific aspect of the infrastructure (e.g., the availability status of a service).
- S_i denotes the surveillance mechanism representing the monitoring method employed (e.g., Auditd rule, individual polling service)
- L_i denotes the set of logs (i.e. a textual, abstract entry in a file, resulting from an event or action) generated as a result of the surveillance mechanism S_i.

The relationship between the surveillance mechanism S_i and set of logs L_i can be formally expressed as $L_i = f(S_i)$, where $f(S_i)$ represents the utilization of the surveillance mechanism S_i resulting in the generated logs L_i.

3.2.2 Relationships Between Monitoring Points

Monitoring points can be related to each other in two ways:

- *Aggregation* means that activities from different monitoring points (of type "Activities") need to be consolidated to provide a unified view of participant actions. This aggregation process can be represented by a function called Agg.
- *Influence* means that a monitoring point (or an aggregation of monitoring points) collectively affects the status change of another monitoring point (of type "State"). This influence can be captured by a function called Inf that describes how the aggregated activities influence the status of another point.

Let MP_i denote monitoring points, where $i \in \{1, 2, ..., n\}$.

The aggregation of activities from multiple monitoring points of type "Activities" can be represented as follows:

$$MP_1, MP_2, ..., MP_n \xrightarrow{Agg} MP_{\text{aggregated}} \quad (1)$$

In this example, activities from n different monitoring points of type "Activities", denoted by $MP_1, MP_2, ..., MP_n$, are aggregated to form $MP_{\text{aggregated}}$. Single monitoring points, or the aggregation of monitoring points may then influence the status of another monitoring point, which is expressed as follows:

$$MP_{agg} \xrightarrow{Inf} MP_j \quad (2)$$

Here, $MP_{\text{aggregated}}$ represents the aggregated activities of multiple monitoring points, which then influence the status of monitoring point MP_j.

The specific form of functions Agg and Inf, as well as the mechanisms for aggregation and influence, may vary depending on the specific requirements and characteristics of the system.

3.3 Realising Monitoring Points

The approach to implementing monitoring points consists of four steps, which are illustrated in Fig. 1 and described in more details in the following subsections.

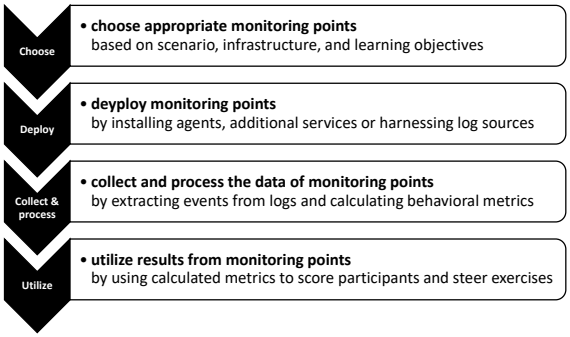

Figure 1: Approach to implementing monitoring points.

3.3.1 Choose Appropriate Monitoring Points

A prerequisite for selecting appropriate monitoring points is that the cyber exercise scenario, the tasks and learning objectives therein, and the associated infrastructure are already defined. This information provides a basis that allows for determining what data should be obtained from the infrastructure and how it should be processed (e.g., determining whether a specific task has been solved for scenario control, determining how specific tasks were solved for feedback provisioning). Additionally, it should be determined how the desired data will be obtained (e.g., through audit logs, through application logs, through individual observer services). It is not always possible to yield the desired results with a single monitoring point. Therefore, at this stage, considerations should be made about how monitoring points can be related to each other. Specifically, which monitoring points should be aggregated or which ones represent influencing factors for other monitoring points.

3.3.2 Deploy Monitoring Points

When deploying monitoring points, the characteristics defined in the stage before are implemented on the cyber exercise infrastructure. This involves either installing and configuring existing services or developing and deploying new ones. Additionally, it is important to carefully consider the risks associated with monitoring points and, if necessary, align them with the scenario. On one hand, monitoring points may provide clues to solution approaches (if participants discover the running services), prompting the need to restrict permissions. On the other hand, monitoring points could generate conflicts that disrupt the exercise scenario. For example, if the exercise task involves forensic examination of a host, caution should be exercised to avoid creating misleading distractions through the monitoring points.

3.3.3 Collect and Process the Data of Monitoring Points

During the exercise, monitoring points collect data and store it in the form of logs. To extract desired information from the generated logs, they need to be filtered and analyzed. This process allows for the calculation of metrics, providing insights into participant activities and performance. Additionally, data analysis techniques can be applied to identify patterns, trends, and anomalies, further enhancing the understanding of exercise dynamics and participant behavior.

3.3.4 Utilizing Results from Monitoring Points

After the data has been collected and processed, it serves as a basis for the calculation of metrics and their utilization. We propose to categorize the resulting metrics into the following three categories:

- *Speed / Duration.* Metrics in this category measure the elapsed time until a certain state is reached in the system (for example, the speed with which a specific task has been solved) or the time spans certain states have lasted (for example, the duration of the availability of a certain service).
- *Efficiency.* Efficiency metrics measure how efficiently participants have reached a certain state in the system. This includes calculating the number of commands and their complexity used to solve a specific task or the number of attempts required for a particular task.
- *Compliance.* Compliance metrics capture, similar to efficiency metrics, the path participants have taken to reach a certain state. However, the focus here is not on efficiency but on compliance with procedures. Often numerous ways exist to reach a certain state, however only some are compliant to given guidelines or specifications. For example, these metrics may be applied to determine whether participants used option X, compliant to company standards, or alternative option Y in order to reach a certain result.

The data collected through monitoring points, as well as the calculated metrics, can serve multiple purposes, enabling a deeper understanding and optimization of cyber exercises: These purposes include:

- *Scoring.* Assessing participant performance by evaluating their actions and decisions during the exercise (e.g. participants gain points for speed, efficiency or compliance to a certain baseline).
- *Providing Feedback to Participants.* Providing participants with insights into the paths they chose, highlighting areas for improvement and offering suggestions for other approaches.
- *Scenario Control.* Using real-time metrics during the exercise to dynamically adjust the scenario. For instance, participants who quickly solve challenging tasks can be presented with additional or more difficult challenges, while those struggling may receive adapted tasks to maintain engagement and learning.
- *Optimization of the Exercise Environment.* Drawing insights for future exercises by identifying what worked well, what needs improvement, and how the overall infrastructure can be enhanced for better outcomes.

4 PROOF-OF-CONCEPT IMPLEMENTATION

To demonstrate the practical applicability of the monitoring points introduced in Sect. 3, we developed a proof-of-concept implementation and evaluated it in the context of a case study. As part of this evaluation, a cyber exercise was conducted in which participants were tasked with solving five specific challenges. These challenges provided the basis for showcasing and evaluating the applicability and effectiveness of the proposed monitoring points.

4.1 Technical Preparations

The cyber exercise was implemented on the AIT Cyber Range (Leitner et al., 2020), where each participant had access to a virtual machine pre-installed and pre-configured with all necessary technical components for both conducting the exercise and utilizing the monitoring points. The virtual machines were running on the Linux Ubuntu 20.04 LTS operating system with a Mate desktop environment. Additionally, we set up a noVNC server and installed VNC on the participant machines to enable remote desktop connections to the machine through any web browser. To conduct the exercises, only the standard installation of an Apache2 web server and the configuration of certificates (for the use of HTTPS without warnings) were required, which was performed using Ansible.

In addition to configuring the participant machines, we used the Web Platform *Learners* (Reuter et al., 2023), to provide participants with a possibility to authenticate themselves and gain access to the client machine via noVNC (see the upper part of Figure 2), as well as to distribute information, documentation and tasks. During the exercise, we transmitted the cyber exercise tasks to participants utilizing the *Learners* Web Platform and created a submission page that allowed participants to send a confirmation once they have successfully completed a task.

4.2 Cyber Exercise Scenario

As the participants were interested in IT topics, but merely had only basic knowledge in computer science and cybersecurity, a straightforward and well-explained scenario including five challenges was chosen. The task descriptions additionally included possible solution paths to provide participants with guidance, if necessary. Prior to conducting the monitored cyber exercise scenario, a training session on basic Linux Bash skills was conducted to equip participants with fundamental knowledge. Additionally, a cheat-sheet for basic Linux commands was provided, particularly for those participants with no prior experience.

The cyber exercise scenario comprised five tasks related to the configuration of an Apache2 web server. Following, the specific tasks are explained:

1. **Replace Index Page.** Originally, the default Apache2 web page was the index page (index.html). Additionally, another HTML page named example.html was provided in the same directory. The task was to rename or remove the default Apache2 index page and rename the example.html page to index.html. The task is considered successfully completed when the new page is displayed instead of the default Apache2 webpage upon accessing localhost in the browser.
2. **Forward HTTP to HTTPS.** In the exercise infrastructure, valid certificates were already prepared, and the localhost page was accessible both under HTTP and HTTPS (with a valid certificate). The task was to configure the web server, that network traffic arriving on port 80 (HTTP) is automatically redirected to port 443 (HTTPS). The task was considered completed when accessing http://localhost in the browser automatically redirects to https://localhost.
3. **Deactivate Directory Listing.** On the web server, there was a directory named ”secret” that, upon accessing (localhost/secret) via browser, displayed the files it contained (directory listing).

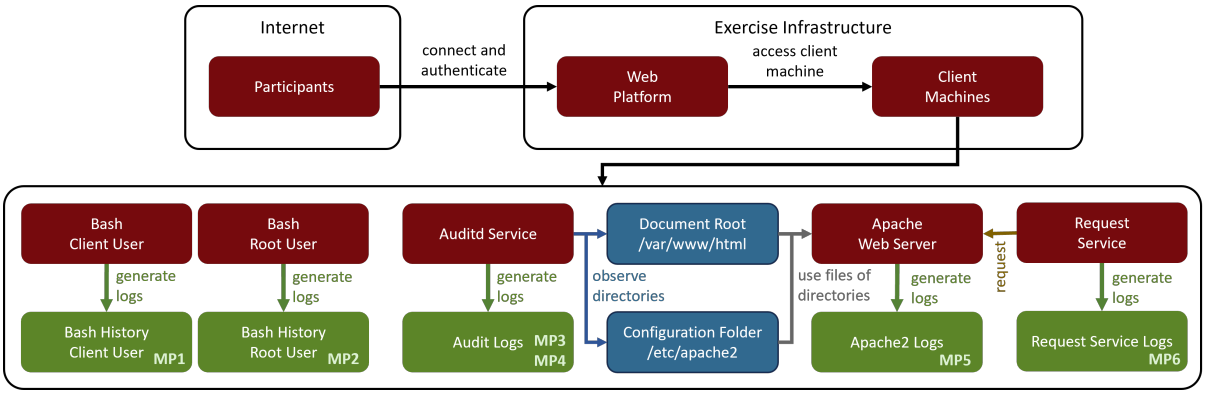

Figure 2: Infrastructure overview.

The task was to configure the web server in a way, that the contents of the folder are no longer listed. The task was considered completed when accessing localhost/secret in the browser no longer listed any files from the directory.

4. **Deactivate File Access.** In this task, access to a specific file needed to be restricted. Within the ”secret” directory a file named ”secret.txt” was accessible. The objective was to configure the web server to prohibit access to this file. The task was considered completed, when accessing localhost/secret/secret.txt returns a permission denied information.
5. **Change Folder Permissions.** The directory /var/www/html, which served as the document root for the web server, was owned by the root user. The task was to change the owner of the directory to the www-data user. The task was considered completed when the owner of the directory was changed accordingly and the website remained accessible in the browser.

The tasks were triggered sequentially in the specified order during the exercise. Each task was only unlocked after the previous one was completed by all participants to ensure clear start and end times for evaluation purposes. Each task took approximately 5-10 minutes to complete. Participants were instructed to solve the task and, once they believed they had completed it, to confirm their completion on the *Learners* Web Platform. This allowed us to compare the time at which participants actually completed a certain task with the time at which the monitoring points detected the completion of that specific task.

4.3 Monitoring Points

To detect the activities of the participants in the cyber exercise utilizing this proof-of-concept implementation, the following monitoring points were configured or identified:

1. **Bash History Client.** Since participants mostly worked from the terminal, the default Bash history represents a good means to gather information. To enhance its detail, the existing standard history was augmented with a timestamp to enable temporal inference. To achieve this, the line `export HISTTIMEFORMAT="%F %T"` was added to the file ~/.bashrc of the client users.
2. **Bash History Root.** To also capture the Bash history including timestamps when the participant executes commands as the root user, the same configuration was applied to the root user as well.
3. **Audit Rule /etc/apache2.** Since /etc/apache2 is the configuration folder for the Apache2 installation, some of the tasks involve actions within this directory. Therefore, the following Auditd rule was defined to monitor the /etc/apache2 folder: `auditctl -w /etc/apache2/ -p rwxa -k key_apache2` – where `-w` specifies the directory to be monitored, `-p` defines the access operations (i.e. read, write, execute, attributes), and `-k` defines a key to associate logs with this rule. This rule will generate audit logs whenever there is any change or access (read, write, execute, attributes) to files within the Apache configuration directory /etc/apache2/, enabling tracking and investigation of such activities.
4. **Audit Rule /var/www/html.** The same Auditd rule as for the configuration folder was also applied to the Document Root folder /var/www/html: `auditctl -w /var/www/html/ -p rwxa -k key_html`.
5. **Apache2 Default Logs.** Since all tasks involve configurations on an Apache2 web server or its directories, the Application Logs of the Apache2 web server are relevant for tracking corresponding activities. No configurations were made for

this; instead, the default log settings were used as a monitoring point.

6. **Individual Request Service.** All the aforementioned monitoring points observe the behavior of participants on the system, resulting in application logs (e.g., Apache2), Bash histories, or logs of changes to the filesystem. However, to determine whether changes actually work and the pages on the web server are accessible accordingly, an additional individual request service was developed. This is a simple Python3 service that regularly (in our case, every 20 seconds) requests the web pages relevant to the tasks and logs the results (already pre-filtered) in a log file.

Figure 2 gives an overview of the entire cyber exercise infrastructure and puts special emphasis on the monitoring points deployed on client machines. Within the client machine, the green boxes represent the resulting logs of monitoring points (MP), while the red boxes represent the services generating the logs. In case of the Auditd service, the blue boxes represent the directories observed by the Auditd rules.

4.4 Interpretation of Monitoring Points

While monitoring points (1) to (5) observe changes in the filesystem or in applications, monitoring point (6) simply continuously checks and logs the http status code returned by the web server for a specific webpage. These status codes allow for conclusions whether tasks were solved or not - it merely determines the status of the web server through continuous requests and is thus of type ”State”. However, monitoring points (1) to (5) determine activities (e.g., changes to files, configurations, etc.) that participants perform on the system. They are therefore of type ”Activities”.

Hence, Monitoring Points (1) - (5) are in an aggregating relationship, as each represents activities that complement each other through their aggregation. The aggregation of Monitoring Points (1) to (5) stands in an influencing relationship with Monitoring Point (6). In other words, the activities observed in Monitoring Points (1) - (5) influence the state observed in Monitoring Point (6).

The type and relationships of monitoring points is important for their interpretation. Monitoring Points (1) - (5), classified as ”Activities,” observe activities. As mentioned in the relationship, their logs need to be aggregated to obtain a complete picture of the activities. The observed activities from a single monitoring point may be incomplete. Furthermore, relying solely on observing activities makes it difficult to draw reliable conclusions about whether tasks have been completed or not (i.e., a certain expected state has been reached). For this purpose, the state-observing Monitoring Point (6) is required (influencing relationship).

This example illustrates the implementation of Monitoring Point 6 and its application for evaluation using speed metrics in relation to Task 3 ”Deactivate Directory Listing”. Monitoring Point 6 was implemented through a simple, individual Request Service that periodically checks whether the directory listing on the local web server is activated, as shown in the following example:

```
while True:
  resp = requests.get("https://localhost/secret")
  log.write(f"{datetime.now()}:{resp.status_code}\n")
  time.sleep(20)  # configure interval
```

During the exercise, this service runs, and logs are generated. Following, there is an excerpt of logs from one participant in the exercise. As seen, the status code changes from 200 OK to 403 Forbidden at 09:30:21. Thus, this is the point in time when the task is considered ”solved.” Now, a script can search the log file for the first occurrence of the status code 403, calculate the difference from the start of the exercise, and thus obtain a speed metric determining how long a participant took for an exercise.

```
2024-03-01 09:29:41: 200
2024-03-01 09:30:01: 200
2024-03-01 09:30:21: 403
2024-03-01 09:30:41: 403
```

The exercise started at 09:27:00. Therefore, this participant took 03:21 minutes to solve the task. Notice, that the accuracy of the calculation depends on the configured polling interval of the Request Service (here 20 seconds).

Figure 3: Example of utilizing a monitoring point to calculating a speed metric.

The selected monitoring points enable metrics to be generated for all categories mentioned in Sect. 3.3.4. Monitoring Point (6) allows for the determination of when tasks were solved by regularly querying the status (Speed/Duration). Monitoring Points (1)-(5) enable the detection of activities on the systems that lead to status changes. This allows for the creation of both efficiency metrics and compliance metrics.

Figure 3 illustrates a complete example of utilizing a monitoring point to calculating a speed metric.

5 RESULTS AND DISCUSSION

The proof-of-concept implementation (Sect. 4) was applied in a cyber exercise case study comprising 14 participants. The following subsections describe the

main results and the related discussion of the case study.

5.1 Participant Profile

The 14 participants have all recently completed a higher education and are generally interested in IT topics. To assess the participants' knowledge level concerning the exercise, five relevant questions were posed, querying their experiences with the following topics: (1) the Linux operating system, (2) the Linux Bash, (3) service configuration, (4) the web protocols HTTP and HTTPS, and (5) setting up a website. Participants were asked to self-assess their knowledge in these areas on a scale ranging from None, Beginner, Intermediate, to Advanced. Overall, it was observed that there were varying levels of knowledge among the participants, with the average indicating either none or little experience in the specified areas. Table 1 shows the details of this self assessment per skill.

5.2 Exercise Durations

As depicted in Sect. 4.2, five tasks were addressed in the exercise. The precise duration of each task was slightly varied based on the number of participants who had reported completing the task. Table 2 illustrates the exact duration for each of the five exercises along with their start and end times. On average, each task lasted 7 minutes and 24 seconds.

5.3 Exercise Results

Using Task 2 'Forward HTTP to HTTPS', as an example, we demonstrate how a metric can be calculated and utilized for the categories *Speed/Duration*, *Efficiency*, and *Compliance*, as outlined in Sect. 3.3.4. Table 3 displays the 14 participants (unranked, in random order), the timestamp when the task was recognized as being solved, and the results of the calculated metrics for each category. Subsequently, we explain the meaning of each metric and how it was calculated.

1. *Speed / Duration.* The metric of type Speed/Duration represents the elapsed time from the start of the exercise at 09:16 (see Table 2, Task 2) until the task was solved. The moment when the task was solved was determined by Monitoring Point (6) - the individual Request Service (explanations provided in the breakout box in Figure 3). In general, the result can be interpreted as follows: the faster a participant solves the task, the better their performance.
2. *Efficiency.* The Efficiency metric in this example is measured by the number of Bash commands used from the start of the exercise until the point when the Request Service reported the completion of the exercise. For this purpose, the Bash history of the client user was aggregated with the Bash history of the root user, as participants could also switch to the root user using `sudo -i` and execute commands as root. This aggregation affected the result measured by Monitoring Point (6), so only the number of commands executed until the observed status change by Monitoring Point (6) was measured. Formally, the relationship between these points can be represented as follows: $MP_j = Agg(MP_{bash}, MP_{root})$. Some participants solved the task after the exercise had already ended (as other tasks had been solved in the meantime). Therefore, rendering the number of commands for these participants does not give any indication on how many commands were needed for the specific task and thus were removed from further considerations.
3. *Compliance.* The task could be solved in several different ways, with two of them explained to the participants beforehand. The first method involved creating a Virtual Host for port 80 (HTTP) in the configuration file '000-default.conf' to automatically redirect traffic on this port to port 443 (HTTPS). The other option was to deactivate the web server listening on port 80 in the 'ports.conf' configuration file, thereby automatically redirecting traffic from port 80 to port 443. To determine how participants solved the task, Monitoring Point 4 was utilized, which generates audit logs for actions in the /etc/apache2 directory. The audit logs were filtered and specifically searched to see whether the '000-default.conf' file or the 'ports.conf' file was edited. Consequently, it was defined which of these two methods the participant actions were compliant with. As shown in Table 3, 12 participants solved the task using the '000-default.conf' file, one participant solved it using the 'ports.conf' file, and one participant was unable to solve the task.

5.4 Discussion of Results

The results of our case study demonstrate that the metrics generated from the monitoring points optimize the observability of cyber exercises, providing exercise management with more targeted information on how participants behave in the exercise environment. This opens up extensive possibilities for fundamentally improving cyber exercises. In the follow-

Table 1: Participants' Experience.

Skill	None	Beg	Int	Adv	Median
Linux OS	7	4	2	1	**None / Beg**
Linux Bash	7	4	2	1	**None / Beg**
Service Configuration	7	4	2	1	**None / Beg**
HTTP / HTTPS	4	5	3	2	**Beg**
Set up Website	4	7	1	2	**Beg**

Table 2: Duration of the tasks.

Nr	Task	Time interval
1	Replace Index Page	09:07 - 09:15
2	Forward HTTP to HTTPS	09:16 - 09:25
3	Deactivate Directory Listing	09:27 - 09:33
4	Deactivate File Access	09:35 - 09:44
5	Change Folder Permissions	09:44 - 09:49

ing subsections, therefore, several points will be discussed (based on the metrics calculated in Table 3).

5.4.1 Scoring

The calculated metrics enable the design of efficient scoring mechanisms for cyber exercises, allowing for the evaluation of participants. Points can be automatically awarded based on the calculated metrics. In our context, this could mean, for example, that a faster solution of the task results in more points; more points for a more efficient use of commands (lower efficiency count); or more points for using a specific path (for example, solving the task by editing the 'ports.conf' file) because fewer participants used this path, or because it is defined beforehand how many points participants get for certain paths.

5.4.2 Providing Feedback for Participants

The results of the proof-of-concept implementation presented in this paper allow conclusions to be drawn about how participants performed in an exercise. This is very helpful for providing goal-oriented and high-quality feedback to the participants. Especially the compliance metrics are suitable for addressing in the feedback which solution path participants used, and what other, potentially more efficient solutions could have been possible.

5.4.3 Scenario Control

In the current implementation, all evaluations were conducted after the exercise. However, it is conceivable to analyze the logs in real time, enabling the scenario to dynamically adapt in more complex exercises. This approach could facilitate the creation of adaptive exercises that adjust to participants' progress. For instance, a participant who quickly completes a task could be presented with subsequent tasks sooner, while those taking longer might receive additional hints. Similarly, participants who solve tasks efficiently could face more challenging tasks or fewer hints in subsequent stages. The progression of the scenario could also vary based on the solution path chosen; for example, participants following the *000-default.conf* path might experience a different sequence of events compared to those using the *ports.conf* path.

5.4.4 Optimization of the Exercise Environment

More detailed insights into participant behavior enable exercise organizers to assess their exercise environment and implement optimization measures for future sessions. If the metrics reveal that participants face significant difficulties with certain tasks or follow unexpected solution paths, this may suggest a need for organizers to offer additional hints or clearer explanations.

6 CONCLUSION AND FUTURE WORK

This paper introduced the concept of monitoring points and demonstrated their application through a proof-of-concept implementation. Monitoring points are strategically positioned observation mechanisms within a cyber exercise infrastructure, designed to provide targeted insights into participant activities and the status of system components, even in complex environments. By aggregating and correlating data from multiple points across various servers or components, monitoring points enable the creation of a comprehensive view of participants' actions, allowing for a deeper understanding of how tasks were approached and solutions achieved.

Metrics such as Speed/Duration, Efficiency, and Compliance serve as illustrative examples of how the collected data can be utilized. They allow for the assessment of how quickly participants completed tasks, how efficient their solutions were, and whether

Table 3: Calculated Metrics.

Nr	End time	Speed/Duration	Efficiency	Compliance
1	09:21:59	05:59	16	000-default.conf
2	not finished	-	11	-
3	09:23:50	07:50	17	000-default.conf
4	09:25:06	09:06	(5)	000-default.conf
5	09:28:45	12:45	-	000-default.conf
6	09:17:31	01:31	3	000-default.conf
7	09:18:37	02:37	9	ports.conf
8	09:20:18	04:18	9	000-default.conf
9	09:19:28	03:28	7	000-default.conf
10	09:21:20	05:20	18	000-default.conf
11	09:43:14	17:14	-	000-default.conf
12	09:26:35	10:35	-	000-default.conf
13	09:25:17	09:17	(14)	000-default.conf
14	09:22:32	06:32	10	000-default.conf

their chosen solution paths adhered to predefined baselines. The calculated metrics form the foundation for efficiently evaluating participants, delivering high-quality feedback, dynamically adjusting scenario control, and optimizing the exercise infrastructure. These capabilities lay the groundwork for a more advanced understanding of participant behavior and support the development of adaptable and robust cyber exercises.

Future work will focus on applying the introduced concept of monitoring points in a live exercise environment to evaluate their practicality for calculating metrics, guiding inject delivery, and facilitating adaptive exercise design. This will involve a comprehensive examination of how these metrics can be effectively utilized to assess participants' progress during exercises, enabling dynamic adjustments based on real-time data insights. By integrating these metrics into the exercise framework, the goal is to enhance the exercise's responsiveness to evolving threats and challenges, ultimately creating a more adaptive and effective cybersecurity training environment.

ACKNOWLEDGEMENTS

This work was partly funded by the Austrian Research Promotion Agency (FFG) project STATURE (grant no. FO999888556) and by the Digital Europe Programme (DIGITAL) project CYBERUNITY (grant no. 101128024).

REFERENCES

Abbott, R. G., McClain, J., Anderson, B., Nauer, K., Silva, A., and Forsythe, C. (2015). Log analysis of cyber security training exercises. *Procedia Manufacturing*, 3:5088–5094.

Andreolini, M., Colacino, V. G., Colajanni, M., and Marchetti, M. (2020). A Framework for the Evaluation of Trainee Performance in Cyber Range Exercises. *Mobile Networks and Applications*, 25(1):236–247.

Braghin, C., Cimato, S., Damiani, E., Frati, F., Riccobene, E., and Astaneh, S. (2020). Towards the Monitoring and Evaluation of Trainees' Activities in Cyber Ranges. In Hatzivasilis, G. and Ioannidis, S., editors, *Model-driven Simulation and Training Environments for Cybersecurity*, Lecture Notes in Computer Science, pages 79–91, Cham. Springer International Publishing.

Čeleda, P., Čegan, J., Vykopal, J., Tovarňák, D., et al. (2015). Kypo–a platform for cyber defence exercises. *M&S Support to Operational Tasks Including War Gaming, Logistics, Cyber Defence. NATO Science and Technology Organization*.

Eagle, C. (2013). Computer security competitions: Expanding educational outcomes. *IEEE Security & Privacy*, 11(4):69–71.

Henshel, D. S., Deckard, G. M., Lufkin, B., Buchler, N., Hoffman, B., Rajivan, P., and Collman, S. (2016). Predicting proficiency in cyber defense team exercises. In *MILCOM 2016 - 2016 IEEE Military Communications Conference*, pages 776–781. ISSN: 2155-7586.

Karjalainen, M., Kokkonen, T., and Puuska, S. (2019). Pedagogical aspects of cyber security exercises. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 103–108. IEEE.

Kucek, S. and 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.

Labuschagne, W. A. and Grobler, M. (2017). Developing a capability to classify technical skill levels within a cyber range. In *16th European conference on cyber warfare and security, ECCWS 2017*, pages 224–234.

Leitner, M., Frank, M., Hotwagner, W., Langner, G., Maurhart, O., Pahi, T., Reuter, L., Skopik, F., Smith, P., and Warum, M. (2020). Ait cyber range: flexible cyber security environment for exercises, training and research. In *Proceedings of the European Interdisciplinary Cybersecurity Conference*, pages 1–6.

Li, Y. and Xie, M. (2016). Platoon: A virtual platform for team-oriented cybersecurity training and exercises. In *Proceedings of the 17th Annual Conference on Information Technology Education*, pages 20–25.

Macak, M., Oslejsek, R., and Buhnova, B. (2022). Applying Process Discovery to Cybersecurity Training: An Experience Report. In *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 394–402. ISSN: 2768-0657.

Maennel, K. (2020). Learning analytics perspective: Evidencing learning from digital datasets in cybersecurity exercises. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 27–36. IEEE.

Mirkovic, J., Aggarwal, A., Weinman, D., Lepe, P., Mache, J., and Weiss, R. (2020). Using Terminal Histories to Monitor Student Progress on Hands-on Exercises. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 866–872. Association for Computing Machinery, New York, NY, USA.

Patriciu, V.-V. and Furtuna, A. C. (2009). Guide for designing cyber security exercises. In *Proceedings of the 8th WSEAS International Conference on E-Activities and information security and privacy*, pages 172–177. World Scientific and Engineering Academy and Society (WSEAS).

Red Hat, I. (2022). *Linux Audit Framework*. Red Hat, Inc. `https://linux.die.net/man/8/auditd`.

Reuter, L., Akhras, B., Allison, D., Agron, B., Hewes, M., Paulino Marques, R., Soro, F., and Smith, P. (2023). Supporting flexible and engaging computer security training courses with the online learners platform and hands-on exercises. In *IAEA International Conference on Computer Security in the Nuclear World: Security for Safety*.

Snyder, R. (2006). Combining an e-commerce simulation with a cyber-survivor exercise. In *Proceedings of the 3rd annual conference on Information security curriculum development*, pages 92–95.

Vinlove, Q., Mache, J., and Weiss, R. (2020). Predicting student success in cybersecurity exercises with a support vector classifier. *Journal of Computing Sciences in Colleges*, 36(1):26–34.

Vykopal, J., Ošlejšek, R., Burská, K., and Zákopčanová, K. (2018). Timely feedback in unstructured cybersecurity exercises. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 173–178.

Vykopal, J., Vizvary, M., Oslejsek, R., Celeda, P., and Tovarnak, D. (2017). Lessons learned from complex hands-on defence exercises in a cyber range. In *2017 IEEE Frontiers in Education Conference (FIE)*, pages 1–8.

Weiss, R., Locasto, M. E., and Mache, J. (2016). A reflective approach to assessing student performance in cybersecurity exercises. In *Proceedings of the 47th ACM technical symposium on computing science education*, pages 597–602.

Wen, S.-F., Yamin, M. M., and Katt, B. (2021). Ontology-based scenario modeling for cyber security exercise. In *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 249–258. IEEE.

Yamin, M. M., Katt, B., Torseth, E., Gkioulos, V., and Kowalski, S. J. (2018). Make it and break it: An iot smart home testbed case study. In *Proceedings of the 2nd International Symposium on Computer Science and Intelligent Control*, pages 1–6.