International Journal of Information Security (2025) 24:96
https://doi.org/10.1007/s10207-025-00993-6

REGULAR CONTRIBUTION

Exploring a comprehensive approach to customize cyber exercises utilizing a process-based lifecycle model

Tobias Pfaller[1] · Florian Skopik[1] · Paul Smith[2] · Maria Leitner[3]

Abstract

Cyber exercises enable the effective training of cyber security skills in a simulated, yet realistic, environment for a wide variety of professional roles. However, planning, conducting, and evaluating customized (i.e., non-standard) cyber exercise scenarios involves numerous time- and resource-intensive activities, which are still mostly carried out manually today. Unfortunately, the high costs related to these activities limit the practical applicability of cyber exercises to serve widely as a regular tool for skill development. Cyber exercise scenarios typically involve a sequence of predefined and carefully planned injects (e.g., events) that are rolled out sequentially, driving the progression of the exercise. The composition of such injects resembles a linear process in its simplest form. Therefore, we argue that the utilization of existing, standardized, and well-researched methods from the business process domain provides opportunities to improve the quality of cyber exercises and at the same time reduce the workload necessary for planning and conducting them. This paper reviews the challenges related to conducting customized cyber exercises and introduces a process-based cyber exercise lifecycle model that leverages the power of process modeling languages, process engines, and process evaluation methods / metrics to transform cyber exercises into transparent, dynamic, and highly automated endeavors. Therefore, the approach presented utilizes process modeling to plan cyber exercise scenario in a structured and flexible manner, enabling the creation of dynamic paths that adapt to participants' actions. These process models are directly executed by process engines, which automate the rollout of injects and collect detailed logs for evaluation purposes. We further describe the application of this lifecycle model in course of a proof-of-concept implementation and discuss technical insights as well as lessons learned from its utilization at a large-scale national cyber exercise together with CERTs and authorities. While the state of the art mostly focuses on optimizing individual tasks or phases within the cyber exercise lifecycle, our contribution aims to offer a comprehensive integrated framework that spans across the phases, providing interfaces between them, and enhancing the overall effectiveness and maintainability of cyber exercises. Further, we are discussing implications of using our approach, identifying opportunities for creating interactive cybersecurity training environments, automated feedback mechanisms and interconnected cyber range exercises.

Keywords Cyber exercise · Cyber exercise scenario · Cyber exercise lifecycle · Cyber range · Process engine · Scenario automation

1 Introduction

Many new technological trends, such as cloud computing, virtualization, mobile computing, or the Internet of Things, have emerged in recent years and have substantially changed the way how cyber security is organized today [30]. While in previous times, standard on-premise enterprise networks looked quite similar to each other, this has changed tremendously recently. There are simply too many technological and organizational options to choose from and the increasing complexity of interconnected infrastructures has led to countless individual solutions. Additionally, many traditional

✉ Tobias Pfaller
tobias.pfaller@ait.ac.at

Florian Skopik
florian.skopik@ait.ac.at

Paul Smith
paul.smith@lancaster.ac.uk

Maria Leitner
maria.leitner@ur.de

[1] AIT Austrian Institute of Technology, Vienna, Austria

[2] Lancaster University, Lancaster, UK

[3] University of Regensburg, Regensburg, Germany

Published online: 15 March 2025

industry domains are still being digitalized, resulting in an increasing need to deal with new risks and cyber threats. As a consequence, an almost unmanageable number of new roles were created with flexible, if not entirely diminishing, boundaries between technological areas, organization, and management. While cyber security was long seen as an administrator's job, today, a professional environment demands for appropriate governance and management roles.

The NICE framework [28] is the result of a systematic collection of cyber security roles and their required skills and currently consists of 52 entries. We argue that the emergence of all these new cyber security roles combined with their peculiarities in the different industry domains demand for individual cyber security training to improve incident response capabilities, forensic analysis skills, or simply staff awareness.

Today, a well-known and effective way to acquire and improve vital skills are specialised training activities in course of exercises on cyber ranges [7, 21]. The flow of these cyber exercises usually consists of predefined and meticulously planned injects (e.g. events) that are linearly rolled out to challenge the participants and thus drive the exercise [20]. The composition of such injects resembles a linear process in its simplest form. Therefore, we argue that the utilization of existing, standardized, and well-researched methods from the business process domain provides opportunities to increase the quality of cyber exercises.

We pick the ENISA exercise lifecycle [12], meant as a template for designing, implementing and delivering cyber exercises, as a starting point. This lifecycle provides a common structure and defines phases for identifying relevant aspects of exercises, as well as planning, conducting and evaluating them. Today, cyber exercises are still not widely adopted as they require numerous manual tasks to be carried out, which is costly and resource-intensive. We aim at expanding on ENISA's initial concept and investigating challenges and potential solutions to making cyber exercises more attractive for a larger number of users. In order to achieve this, we investigate means to automate several aspects of cyber exercises over their lifecycle, making them more adaptable, repeatable, customizable as well as effectively deliverable.

Moreover, in the current literature, we have identified three topics, that are particularly relevant in the context of the process-based approach presented in this paper:

1. First, as argued by Yamin and Katt [46], there are significant inefficiencies throughout the entire cyber exercise lifecycle. Many tasks remain manual, with minimal automation, requiring substantial effort to organize and conduct exercises. These inefficiencies are limiting cyber exercises' scalability and accessibility, hindering their potential to regularly serve for skill development and training.
2. Second, there is a strong interest in more accurately and automatically assessing participant behavior to improve feedback mechanisms [1][41][40]. Currently, these assessments are often conducted through manual methods, such as observations, questionnaires or reports. However, manual methods, and especially after-action reports run the risk of becoming reflections targeting on personal interests rather than extracting serious lessons from experience (or observation) [5].
3. Third, effective communication, both with authorities and other relevant entities, is crucial in the case of a cyber security incident. Testing decentralized and cross-organizational structures is essential for improving incident response and preparedness. In this context, there is growing emphasis on optimizing resources and establishing standards for connectivity across European cyber ranges, as highlighted by recent studies [38].

Based on our approach, we will further elaborate on what these three topics mean for the future of cyber exercises, discussing the opportunities it creates for improving efficiency, enhancing feedback mechanisms, and fostering connectivity. To sum up, the contributions of this paper are three-fold:

- *Challenges and Approaches.* This paper reviews the challenges related to conducting customized cyber exercises and introduces a process-based cyber exercise lifecycle model that leverages the power of process modeling languages, process engines, and process evaluation methods / metrics to transform cyber exercises into transparent, dynamic, and highly automated endeavors.
- *Proof-of-Concept Implementation with Case Study.* We further describe insights into technical implementations of an integrated approach for delivering cyber exercises and discuss potential advantages compared to the state of the art. We describe the application of this proof-of-concept in the course of a large-scale national cyber exercise together with CERTs (Computer Emergency Response Teams) and authorities and derive valuable lessons learned.
- *Discussion.* We discuss potential use cases and implications of our approach for future cyber security training. We examine how our process-based lifecycle model can enhance training environments, improve automated feedback mechanisms, and facilitate multi-scenario coordination across diverse infrastructures.

This article is an extended version of the conference proceedings in [29]. The remainder of the paper is organized as follows. Section 2 outlines important background and related work. Then, Sect. 3 summarizes the identified

challenges as well as potential solutions from the literature. Section 4 describes a proof-of-concept (PoC) implementation of some of the outlined solutions that together realize an integrated version of ENISA's lifecycle model for cyber exercises. We applied this PoC in the course of a case study, which is described in Sect. 5, and discuss major findings. Sect. 6 discusses potential applications of our approach and implications for the future of cyber security training. Finally, Sect. 7 concludes the paper.

2 Background and related work

Cyber exercises can take place in a simulation environment that allows for training and testing cyber security capabilities. In that context, cyber security capabilities comprise anything that contribute to an improved cyber security posture, as for example, technical skills and cyber security awareness from an individual's perspective or incident response processes and communication structures from an organisational perspective. Depending on which capabilities should be trained, cyber exercises vary regarding their type and can be classified into three main types:

1. *Discussion-based exercises* (i.e. table-top exercises [2, 20]) focus on scenario-based discussions, where participants collaboratively explore different scenarios and decision-making processes to address potential cyber incidents. These exercises emphasize strategic and organizational aspects, such as refining communication protocols and incident response plans.
2. *Technical exercises* [18, 33, 44] concentrate on hands-on training to enhance specific technical skills and actions. These exercises often involve realistic technical simulations and tasks, such as identifying and mitigating vulnerabilities, responding to active attacks, or securing systems under stress.
3. *Hybrid exercises* [6, 8, 39] combine elements of both technical and discussion-based exercises. These exercises integrate technical actions, such as incident mitigation or system recovery, with organizational and administrative components, including management-level decision-making, inter-team communication, and overall incident coordination. Hybrid forms aim to provide a comprehensive training experience that bridges technical and organizational cybersecurity challenges.

The core of every cyber exercise consists of a (typically) fictional, yet realistic, scenario. The scenario drives the cyber exercise and defines the environment and roles into which the participants immerse themselves. According to Wen et al. [42] a cyber exercise scenario consists of a (1) *Scenario Information Model*, that contains high-level information about the scenario topic, a (2) *Scenario Operation Model* that represents the exercise infrastructure, and the flow of injects, and a (3) *Security Knowledge Model* that contains the security knowledge that should be conveyed to participants. Cyber exercises enable to test and train the responses to various cyber incidents within a practical, yet secure and realistic, environment. Therefore, they represent an excellent learning environment for cyber security professionals [15] and can be used to significantly increase their skill-level [16]. However, planning a cyber exercise is a cumbersome process that can last several months [39], in which a great deal of experience, technical know-how, and creativity is required. Furthermore, according to Yamin and Katt [46], existing inefficiencies in the cyber exercise lifecycle hinder the smooth planning, execution and evaluation of cyber exercises, and limit their ability to widely serve as a tool for cyber security skill development.

Fig. 1 ENISA lifecycle (redrawn) [12]

Therefore, some authors have already introduced lifecycles for conducting cyber exercises. Vykopal et al. [39] comprise a cyber exercise into the phases preparation, dry run, execution, evaluation and repetition. They present their experiences on conducting cyber exercises utilizing their lifecycle model. Furtuna et al. [13] introduce a structured approach for implementing cyber security exercises consisting of seven steps: defining objectives, choosing an approach, technical specifications, creating exercise scenario, establishing a set of rules, choosing metrics (for evaluation), and lessons learned. Seker and Ozbenli [32] present the concept of cyber defence exercises by presenting typical tasks and concepts along the stages planning, execution and evaluation of the ENISA lifecycle [12]. ENISA itself emphasizes the importance of cyber exercises by organizing the Cyber Europe exercise every two years, which serves as a large-scale, cross-border initiative to strengthen international collaboration and test resilience in the face of cyber threats [3, 8]. These exercises are documented through detailed After Action Reports, highlighting key findings, challenges, and improvements.

In this paper, we pick the widely adopted ENISA lifecycle [12] as a basis (see Fig. 1) for our approach.

In the *Identifying* phase of the ENISA lifecycle, the objectives for an exercise are defined, along with initial high-level design decisions (such as exercise type and rough scenario ideas). The *Planning* phase involves detailed planning (such as infrastructure and scenario development, and organizational decisions). The *Conducting* phase encompasses the actual execution of the exercise, and the *Evaluating* phase

involves evaluating the participants as well as the scenario itself and creating final reports.

Researchers already address issues related to certain phases of the lifecycle and contribute associated solutions. When it comes to planning and conducting, Doupé et al. [10], for example, presented a novel cyber exercise design containing missions that were executed utilizing Petri nets. Skopik and Leitner [34] argue that the design of linear cyber exercise scenarios is a simplification that could be problematic in complex cyber exercises. Therefore, they propose an extension with the patterns playback, forking, pause/adapt/repeat, and fast-forward. Leitner et al. [21] are developing the AIT Cyber Range and implemented a scenario engine called GameMaker, that is used to automatically handle a sequential scenario flow and execute non-technical injects, such as participant instructions or email communication. When it comes to evaluation, White et al. [41] argue that simple technical metrics like just measuring if a task was solved in cyber exercises does not give a clear indication of whether the task was fully understood nor whether the learning objectives were achieved. Andreolini et al. [1] developed a framework to model the behavior of participants in a graph. They compare the graphs to those of other participants as well as to a baseline in order to measure performance. White et al. [41] also created graphs from the bash history of participants to get more detailed insights into how they approached different problems they were challenged with.

In particular, the argument of Skopik and Leitner [34], that linear cyber exercise scenarios are a problematic simplification, is highly interesting for our research. The patterns they propose for introducing greater complexity into cyber exercises can also be represented in the form of workflow patterns [35], supported by existing process modeling languages such as BPMN [27]. Consequently, by using a process-like structure to depict complex cyber exercise scenarios, advantages could arise by leveraging Business Process Management (BPM) [43], which offers a comprehensive toolkit of concepts, methods, and techniques for supporting the design, administration, configuration, execution and analysis of processes. This allows for following a BPM lifecycle [11] and utilizing its existing and well-researched tools, including, for example, process engines [23] for automated scenario execution, or process mining [37] for evaluation.

3 Challenges and solutions

To individually tailor cyber exercises to the training objectives and skill levels of different participants or teams, a high degree of flexibility is required, leading to challenges in the planning, conducting, and evaluating phases of the cyber exercise lifecycle (see Fig. 1), which will be explained below.

3.1 Challenges

Cyber exercises are currently executed in a linear fashion, leading to several simplifications that may decrease their realism and thus the learning outcome for participants. As argued by Skopik and Leitner [34], a linear approach is a limiting simplification, and cyber exercises should be designed in a more complex manner to enhance the learning outcomes. Based on current literature, expanded by our experiences, we identified the challenges that currently exist in cyber exercises, as outlined in the rest of this section.

1. *Adaption for different skill levels.* Cyber exercises are extensive training events in which participants of varying skill levels take part [6]. Nevertheless, they adhere to a predefined scenario, which, today, predominantly involves a linear sequence of injects [34]. In a linearly planned exercise, accounting for the individual skill levels of participants proves challenging, leading to persons with high skill levels potentially being underchallenged, while those with low skill levels may be easily overwhelmed. For exercise organizers, it is, therefore, a challenge to make the exercise flexible and adapt it to the progress of each team or participant, thus achieving an appropriate difficulty level for every skill level.
2. *Clear representation of complex scenarios.* The design of cyber exercises is a process involving numerous steps with different tasks and people [25]. When executing the exercise, additional supporters and observers are frequently involved, each of whom must possess a solid view on the scenario. This complexity is further increased when accommodating varying skill levels. Consequently, ensuring a coherent representation of the cyber exercise scenario that enables a comprehensive visualization of the intended flow remains a considerable challenge.
3. *Reusability of parts of a cyber exercise.* The planning and execution of a cyber exercise is a highly time- and resource-intensive task [39]. Due to the rapidly evolving nature of cyber security, typical attack vectors, and vulnerabilities, cyber exercise organizers are challenged to continually develop and improve their scenarios. This challenge is further amplified by the dynamic and ever-changing cyber threat landscape, which requires exercises to adapt to emerging threats by designing realistic, timely, and effective scenarios that reflect current attack strategies. To avoid having to plan a scenario with different learning objectives from scratch each time, it is a challenge to efficiently reuse parts of existing exercises without the need for extensive modifications.
4. *Reduction of complexity during execution.* The execution of a cyber exercise is complex, often erratic, and involves many ad-hoc decisions. Organizers must orchestrate the scenario, which, with an increasing number of teams and

the expanding scope of the scenario, significantly escalates the complexity of the execution [3]. This complexity demands extensive observation and scenario management to stay current and deliver the right inject at the right time. It is a significant challenge to reduce this complexity.

5. *Flexibility during execution.* In a linearly planned cyber exercise, if participants take unexpected actions, it is challenging to react appropriately. It is a significant challenge to make an exercise highly flexible, maintaining some room for maneuver during the execution to be able to respond in an ad-hoc manner to unexpected responses from the participants. Additionally, the linear progression of predefined scenarios often fails to reflect the chaotic and unpredictable nature of real-world cyber incidents [34]. Real incidents frequently involve multiple events unfolding simultaneously or on non-linear timelines, which can make linear exercises feel unrealistic or overly simplified. Ensuring flexibility not only in responding to participant actions but also in adapting scenario progression dynamically is essential for creating exercises that more accurately mirror real-world complexities.
6. *Traceability of exercise processes*. The actions of participants in cyber exercises are often not transparent and are difficult to determine in detail [41]. Often, exercise organizers try to gain a comprehensible insight into the progress of participants through manual observers, questionnaires [16], or reports. All these means cause costs. Increasing traceability by technical means is an important challenge in order to offer participants appropriate, yet cost-effective, feedback.
7. *Individual evaluation.* When the participants' progress and experiences diverge due to their individual pace, a uniform evaluation is not possible. Therefore, individual evaluations must be conducted, which represents a complex and resource-intensive challenge. A further complication arises from the lack of realistic metrics for evaluation. Measuring the success or effectiveness of cybersecurity exercises is often difficult, as desired outcomes-such as improved decision-making, faster response times, and better coordination-are qualitative rather than quantitative. This lack of robust metrics not only complicates individual assessments but also hinders the ability to compare and aggregate results across participants. Developing meaningful, realistic evaluation metrics is thus critical for ensuring accurate assessments of both individual and overall exercise performance.

3.2 Applicable solution

In order to overcome the challenges stated in Sect. 3.1, we propose to harness methods and techniques from business process management to plan, conduct and evaluate cyber exercises. Figure 2 provides an overview of our approach based on the phases of the ENISA lifecycle, extended by process-based concepts that enable seamless transitions between the phases. A process model, which is the output of the planning phase, serves as input for the conducting phase, and exercise logs, that occur during conducting, serve as input for evaluation.

3.2.1 Identifying

According to the ENISA lifecycle [12], the identifying phase consists of identifying the requirements for a cyber exercise and choosing an appropriate exercise type, size and high-level scenario. For this purpose, general methods from requirements engineering [9, 19] are applicable.

3.2.2 Planning

In the planning phase of a cyber exercise, our approach suggests utilizing process modeling languages to model the sequence of injects in a cyber exercise. These languages, such as Business Process Model and Notation (BPMN) [27] or Event-driven Process Chains (EPCs) [17], provide a structured method to visually represent the flow of injects, decisions, and events that occur during the exercise. Process modeling languages allow for a standardized and consistent representation of the exercise's workflow, which is crucial for both planning and execution. They enable exercise planners to identify and document key dependencies, decision points, and alternative scenario paths that may arise during the exercise. This comprehensive modeling helps ensure that all potential outcomes are considered, reducing the risk of oversight and enhancing the overall effectiveness of the exercise. By incorporating process modeling languages into the planning phase, we can overcome the following challenges:

1. *Adaption for different skill levels.* By utilizing workflow patterns provided in process modeling languages, such as BPMN [27] or Event-driven Process Chains (EPCs) [17], complex scenarios with alternative paths can be developed. Consequently, challenging paths can be created for participants with higher skill levels, while additional assistance or even simpler paths can be developed for participants with lower skill levels. Depending on how participants progress during the exercise, a more difficult or easier path is chosen. To adapt a cyber exercise through alternative paths based on participants' progress, this progress must also be measured and evaluated. In our approach, we complement each decision with an indicator that, depending on whether it exceeds certain thresholds or not, determines which specific path is chosen. The values of indicators can be determined, for example, through manual observations or extracted automatically from the

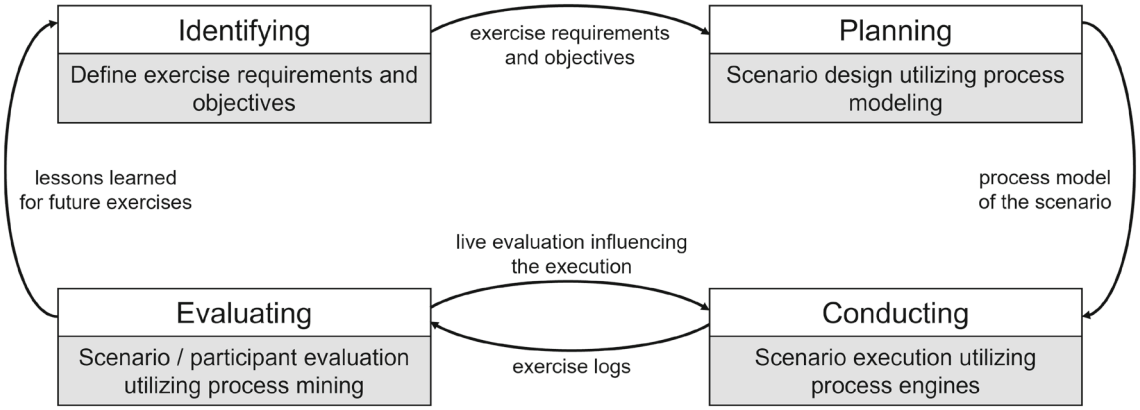

Fig. 2 Process-based cyber exercise lifecycle - overview [29]

observed infrastructure (e.g. log files that capture the participants' actions).

2. *Clear representation of complex scenarios.* The design of complex scenarios in cyber exercises presents significant challenges, particularly in terms of presenting the flow of scenarios in a clear and understandable manner [47]. As exercises increase in complexity, especially when adapting for different skill levels of participants, maintaining a clear representation of the scenario becomes increasingly important. Process modeling languages offer an effective solution to these challenges by enabling a structured and visual representation of the exercise's workflow and allow for leveraging existing standards and guidelines [26]. Stakeholder involved in exercise planning and execution can visually map out the entire sequence of injects, decision points, and potential alternative paths. In traditional representations, such as sequential lists, it is often difficult to discern critical aspects like decisions, dependencies, or parallel actions made by attackers during a simulated scenario. For example, in a simulated Advanced Persistent Threat (APT) attack, the attackers might simultaneously exfiltrate data, deploy ransomware, and establish persistence within the network. Modeling languages like BPMN or EPC can visually represent these parallel actions and the logical flow of the attack, such as a decision to escalate privileges or laterally move to other systems based on network topology. By making these actions and their dependencies explicit, stakeholders gain a clearer understanding of the attack dynamics and can better prepare exercises that simulate realistic threats. Furthermore, the use of process modelling languages promotes effective communication and collaboration among all involved parties, as they ensure that everyone can easily follow the structure and progression of the exercise.
3. *Reusability of parts of a cyber exercise.* Current process modeling standards, such as BPMN support the use of subprocesses [27]. Subprocesses are a powerful concept that allows exercise planners to encapsulate specific parts of a cyber exercise, such as individual attack vectors, into modular components, enhancing the flexibility and reusability of exercise components. Therefore, modular scenarios can be created that can be reused in across different exercises or tailored to various skill levels and objectives. For example, an attack vector represented as a subprocess can be easily integrated into multiple exercise scenarios, allowing it to be reused in different contexts without the need to redesign it from scratch. This modularity not only saves time and resources but also ensures consistency in the training content, as the same subprocess can be applied across various exercises to reinforce specific skills or knowledge areas.

3.2.3 Conducting

After designing cyber exercises using process modeling languages in the planning phase, we utilize process engines during the conducting phase. Process engines provide a dynamic and automated means of executing predefined workflows (i.e. exercise scenarios, in our case) that have been meticulously crafted during the planning phase. By leveraging process engines, the structured sequences of injects and decisions modelled earlier can be seamlessly translated into actionable, real-time exercises. The use of process engines allows for precise control over the timing and sequencing of events, ensuring that the cyber exercise unfolds exactly as designed. This automation not only reduces the manual effort required to coordinate complex scenarios but also enhances the fidelity and consistency of the exercise delivery. Process engines can manage multiple concurrent processes, trigger events based on specific conditions, and adjust the flow dynamically in response to participant actions, making them ideal for managing the intricate choreography of a cyber exercise.

4. *Reduction of complexity during execution.* Specifically, due to the fact that our approach adapts cyber exercises to participants' skill levels and utilizes alternative paths, the complexity of the execution increases significantly. In addition to the already complex task of orchestrating a scenario, decision indicators (a measurable value that serves as input for decisions within the model) must be observed and evaluated, and alternative paths must be considered. Since we already use process modeling languages in the planning phase, the use of a process engine [23] is ideal for overcoming the mentioned challenges. By utilizing a process engine, the process model developed in the planning phase can be instantiated and executed separately for each team and/or participant. Additionally, the values of the decision indicators, can be stored for each team in its own runtime variables. This allows for the automation of the scenario flow, and even complex scenarios that adapt individually to the team's progress, can be executed with a high degree of automation, leading to reduced workload and complexity during execution.
5. *Flexibility during execution.* The use of process engines facilitates a higher degree of flexibility and adaptability in the exercise environment. By planning different scenario paths, a cyber exercise is flexible, and different branches can be chosen based on the participants' progress. That allows to prepare paths for exceptional situations, enabling automatic exercise flow adaptations in such cases. However, in the case of unexpected developments or participant actions that deviate from the expected path, process engines can dynamically adjust the flow of the exercise, introducing new injects or modifying existing ones to steer the scenario back on track. This adaptability is crucial for maintaining the relevance and challenge of the exercise, ensuring that participants are exposed to realistic and evolving cyber threats. Therefore, in case an unforeseen situation arises, where no special path is prepared process engines allow for pausing the scenario execution, adjusting the subsequent elements (injects, decisions, paths, etc.), and afterwards restarting the process from a desired point [24]. Thus, there are no limits to flexibility, and cyber exercise organisers can respond appropriately to unexpected situations. Therefore, the realism and value of the exercise can be enhanced, while participants are effectively prepared for the unpredictable nature of real-world cyber incidents.

3.2.4 Evaluation

Process engines provide robust capabilities for monitoring and logging every aspect of the exercise as it happens. This real-time data collection is invaluable for assessing participant performance, identifying areas where the exercise deviated from the planned scenario, and understanding how different elements of the exercise interact with one another. While monitoring enables the instant detection of bottlenecks or delays in the cyber exercise workflows, allowing for timely interventions, while logs help pinpoint errors or deviations from the expected processes. This facilitates root cause analysis and supports continuous process improvement.

By capturing detailed logs of all activities, process engines lay the groundwork for a comprehensive evaluation in the post-exercise phase, using techniques such as process mining to extract insights and identify opportunities for improvement. The evaluation phase of a cyber exercise can be significantly enhanced by leveraging the detailed event logs and infrastructure artifacts collected during the exercise. The process engine, which orchestrates the delivery of exercise injects and manages the flow of activities, generates comprehensive logs that capture every action taken and decision made throughout the exercise. These insights provide a rich dataset that can be used to (semi-)automatically evaluate both the overall success of the exercise and the specific actions and responses of participants.

6. *Traceability of exercise processes.* Event logs [14] serve as chronological recordings of all events captured within a process flow, providing a detailed trail of the actions and decisions taken throughout a cyber exercise. These logs meticulously document every inject sent, including precise timestamps, decision points, and the values of runtime variables evaluated at or before these decision points. By capturing this information in real-time, event logs enable a high level of traceability and accountability, allowing exercise coordinators to reconstruct the sequence of events and analyze the decision-making processes of participants. Moreover, the traceability provided by event logs facilitates comparative analysis across multiple teams or iterations of the exercise. By analyzing the logs, evaluators can compare how different teams responded to the same scenarios, identifying best practices, common mistakes, or areas where additional training may be needed.
7. *Individual evaluation.* The enhanced traceability afforded by event logs plays a crucial role in monitoring the progress of teams during an exercise. By tracking each step taken by participants, evaluators can assess how well teams are adhering to the exercise plan, identify where they may have deviated from expected actions, and evaluate the effectiveness of their responses to various injects. This detailed tracking provides a clear picture of each team's performance, enabling a more nuanced and individual assessment that goes beyond simply measuring outcomes to understanding the processes that led to those outcomes. The individual information gained from event logs can be utilized to provide teams with detailed feedback on their progress in the exercise. Additionally,

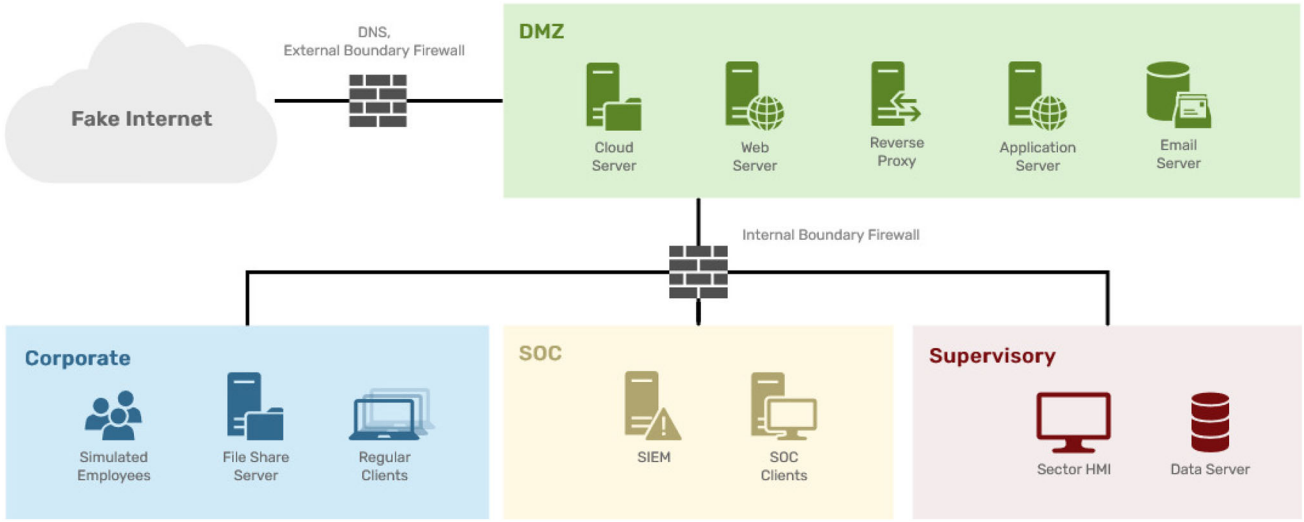

Fig. 3 Infrastructure plan of the proof-of-concept implementation

event logs can serve as input for process mining analyses to identify typical behavioral patterns, positive or negative outliers, or the degree of deviation from a predefined behavioral baseline [37].

4 Proof-of-concept implementation

To demonstrate the practical applicability of our process-based approach, a proof-of-concept implementation was developed and implemented in the context of a national exercise (see Sect. 5). For this purpose, we use a part (i.e., a web defacement attack vector) of the exercise scenario to illustrate the implementation of our concepts within the planning, conducting, and evaluating phases of the ENISA lifecycle. We place particular emphasis on demonstrating how the mechanisms applied in each phase of the lifecycle address the challenges mentioned in Sect. 3.1 and interconnect across phases.

4.1 Exercise setting and infrastructure

In the proof-of-concept implementation of the exercise, participants are divided into teams and take on the role of a SOC (Security Operations Center) team or management personnel. Their task is to manage a generic infrastructure that resembles that of a real company and to respond to anomalies and indicators of compromise both technically and organizationally, as well as through effective communication. The participants' infrastructure setup is shown in Fig. 3. The whole implementation was deployed on the AIT cyber range [21] infrastructure.

In the simulated "Fake Internet," global resources such as mail servers, DNS servers, endpoints accessing public IPs, and attackers are positioned. Behind an external firewall, which also serves as a local DNS server, publicly accessible machines such as cloud servers, web servers, reverse proxies, application servers, and email servers are installed. Behind an internal firewall are the corporate, SOC, and supervisory networks.

The corporate network includes simulated employees who generate realistic network traffic, a file share server for data exchange, and regular clients. These clients are operated by participants who assume management roles and do not have access to the SIEM system. The SOC network houses the SIEM (Security Information and Event Management) system, which aggregates logs from all relevant servers in the infrastructure and displays them in a comprehensive dashboard. This network also contains SOC clients, which are operated by participants and have access to the SIEM system. SOC clients have the rights to connect via SSH to all servers in the network to configure settings and make adjustments.

The supervisory network includes a data server that holds simulated data for a specific sector of the company (e.g., production data, power plant data, etc.) and a Human–Machine Interface (HMI) that visually displays this data on a screen. Participants can connect to their exercise machines via a browser using a NoVNC connection, providing them with a full desktop environment to engage in the exercise.

To maintain the infrastructure and deploy various injects, several management hosts are available within the infrastructure (these are not shown in the diagram, as they are not relevant to the participants' infrastructure). These management hosts have IP addresses in all available networks and a floating IP to be accessible from the Internet, serving as a jump host to the machines within the participants' networks.

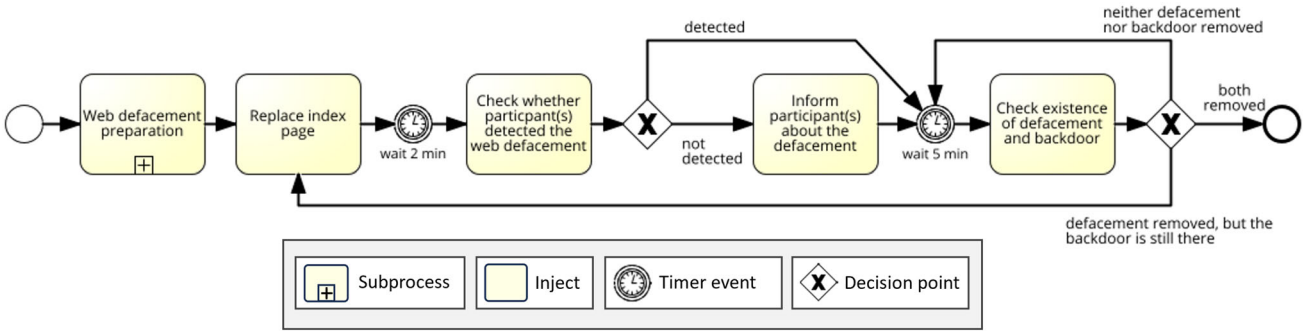

Fig. 4 Representation of the exercise scenario in a process model (BPMN diagram)

This setup allows them to facilitate the deployment of scenario components.

4.2 Planning

Within the planning phase of our cyber exercise, the scenario is meticulously represented through a process model that captures all the essential elements involved. To achieve this, we utilize the widely recognized and standardized process modeling language, BPMN (Business Process Model and Notation) [27].

Figure 4 illustrates the complete implemented model, showcasing how the exercise scenario is translated into a detailed process flow. This model serves as a blueprint for the exercise, outlining the order of injects, the conditions under which specific actions are triggered, and the various pathways participants may follow depending on their responses.

The following sections outline how various BPMN components, such as tasks, subprocesses, timer events, and decisions, are interpreted and applied within the context of a cyber exercise.

4.2.1 Subprocess

Within the model in figure 4, the first task *"Web defacement preparation"* represents a subprocess that encapsulates an attack chain [45], which encompasses a sequence of actions such as scanning, reconnaissance, payload delivery and exploitation. By using a subprocess at this point, the underlying attack vector can be flexibly exchanged, allowing exercise planners to easily swap out different types of attacks without redesigning the entire scenario. This modularity not only enhances the adaptability of the exercise by enabling quick adjustments to reflect new or evolving threats, but it also allows the same subprocess to be integrated into multiple exercise scenarios. Additionally, subprocesses improve traceability by clearly defining and encapsulating distinct phases of an attack chain, making it easier to analyze and document their execution. They also contribute to complexity reduction by breaking down intricate attack vectors into manageable, reusable components, which simplifies both the planning and execution of exercises.

4.2.2 Injects

All tasks in the model comprise injects that serve the purpose of either conveying information or technical artifacts to participants, or requesting specific information from the cyber exercise infrastructure. These injects are strategically placed throughout the scenario to simulate realistic conditions and challenges that participants may encounter in a real-world cyber incident. By delivering timely and relevant information or artifacts, injects guide participants through the exercise, prompting them to respond to evolving situations, make decisions, and execute actions.

4.2.3 Timer events

The timer events (see figure 4) are used to insert waiting times during the scenario, ensuring that participants have adequate time to complete their tasks, offer them a chance to re-assess the situation, or receive feedback on their actions, before continuing with following injects. In a typical process engine, tasks are executed sequentially as soon as the previous one is finished. However, in the context of cyber exercises, this approach is not practical as participants need time to analyze information, make decisions, and take appropriate actions. By incorporating timer events, we can simulate realistic delays and pacing, creating a more authentic and controlled exercise environment that better reflects real-world conditions.

4.2.4 Decisions and decision indicators / runtime variables

The decisions are intended to enable adaptive routing based on the participants' progress. Before each decision, a decision indicator is queried from a runtime variable to utilize its value to decide for certain routes. These runtime vari-

ables play a crucial role in the functioning of the process, as they act as global variables for the duration of a single scenario execution. In the current implementation, runtime variables are stored as strings, enabling them to capture a wide range of information, such as the presence or absence of specific files or conditions. When the process reaches a decision point, these runtime variables can be accessed and their values compared against predefined criteria to determine the appropriate routing for the workflow. For example, a runtime variable storing the status of the index file – whether defaced, replaced, or removed – can be checked, and the process can then branch accordingly to reflect the appropriate next steps. This approach allows for dynamic and data-driven decision-making during the execution of a scenario, ensuring that the process adapts in real-time based on the captured data. By maintaining global accessibility within the scenario execution, runtime variables enable seamless communication between tasks and enhance the flexibility and responsiveness of the workflow.

At the task *"Check whether participant(s) detected the web defacement"*, for example, we aim to determine, based on the web server log files, whether the participant has made a GET call to the website's index page since the defacement was executed. If this has occurred, we assume that the web defacement has been detected, otherwise we assume that it hasn't been detected. This information is stored to a runtime variable and subsequently used for a decision, as stated in the process model. Before the second decision, at the task *"Check existence of defacement and backdoor"*, we inspect the web server's file system (either through automated methods, such as autitd rules or through manual observations) to verify if the defaced index page is still present or has already been replaced, and whether the backdoor that was placed on the system (within the subprocess *"Web defacement preparation"*) is still present or has been removed. To determine if the index page is defaced, several methods can be employed: using auditd rules to monitor changes to the index file (e.g., checking if it is still present, altered, or deleted), conducting a manual observation of the file system, or performing an HTTP request to the web page and analyzing the returned file for signs of defacement. For detecting the backdoor, possible methods include checking for unusual open ports or active connections on the system, utilizing auditd rules to monitor the specific backdoor file for any changes or presence, or conducting a manual inspection of the system for any unexpected processes, files, or configurations indicative of a backdoor. The results of these verification techniques serve as decision indicators and provide the necessary data to guide the subsequent decisions, where three possible routes are available:

1. *Neither the defacement has been removed nor the backdoor deleted:* We will revert and wait for an additional 5 min to allow participants more time.
2. *The defacement has been removed, and the backdoor has been deleted:* The exercise part is finished.
3. *The defacement has been removed, but the backdoor has not been deleted:* The process will return to the *"Replace index page"* task since the attacker still has the opportunity to perform another defacement through the remaining backdoor.

By applying a process modeling language in this example, (1) the scenario is adaptively tailored to the skill level of participants, (2) despite its complexity with various paths and decisions, an understandable and clear model of the scenario is created, and (3) the use of subprocesses ensured the reusability of exercise parts (i.e., an attack vector, in the stated scenario) in other exercises. Moreover, the subsequent conducting phase uses the process model as input for execution, therefore providing a seamless connection between these two phases.

4.3 Conducting

To execute the scenario model developed during the planning phase, we have developed a custom-built process engine called "flowgin". *flowgin* is a lightweight web application built using Flask, utilizing a microservice architecture and leveraging YAML files as input. These YAML files define the flow of given process models and supports the concepts mentioned in the planning phase (i.e., subprocesses, injects, timer events, decisions and decision indicators / runtime variables). Additionally, it has the capability to instantiate multiple processes simultaneously and manage variables during runtime. Flowgin's primary purpose is to manage the flow of injects and does not engage in any other tasks. Exercise administrators access Flowgin through a web-based interface protected by username/password authentication, which provides real-time updates on scenario progress, including elapsed time, completed injects, and upcoming ones. The execution of injects is achieved through HTTP calls. Figure 6 serves as an illustration, showcasing the execution process by highlighting the interaction between the scenario flow executed within the process engine and the participants' exercise environment. For the sake of simplicity, the actual exercise infrastructure has been represented in a simplified manner, featuring only those machines/servers relevant to the website defacement attack vector.

To conduct this scenario individually for each team, flowgin instantiates separate process instances for them. However, since each team possesses its own infrastructure, including web servers, mail servers, attackers (and more) with different IP addresses, users and credentials (e.g. for the mail server login), it is necessary to prefill the model with these variables (e.g. attacker = 172.16.0.32; web-server-ip = 10.0.0.5).

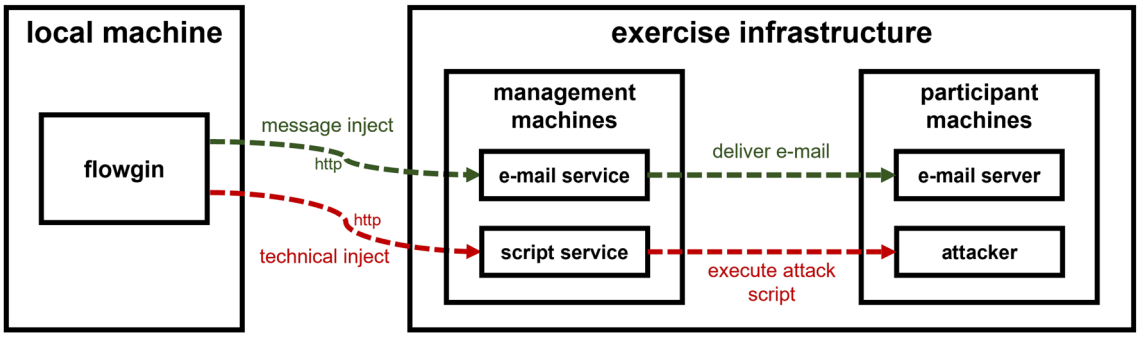

Fig. 5 Delivery of injects from flowgin to the exercise infrastructure

Once all the necessary variables have been provided to the instantiated process, it starts executing and begins with enacting the injects in the specified order. It is important to note that Flowgin operates outside of the infrastructure and makes a REST call for each inject that needs to be delivered into the infrastructure. Therefore, the infrastructure must have externally accessible services that can receive REST requests along with the associated information and then trigger the desired inject accordingly (see Fig. 5).

For message injects, we developed a custom email service that handles information such as the sender, recipient, subject, body, and attachments, and then distributes the email within the infrastructure. This email service can also select from a list of predefined email templates, allowing for the modification of specified content by passing in variables.

For technical injects, we also use a custom-developed tool. It is a lightweight web application that takes HTTP requests and executes scripts based on the provided parameters. This enables the triggering of technical injects, such as simulated attacks, in a flexible and controlled manner.

Figure 5 illustrates the typical execution of injects within our setup. When the process engine triggers an inject, an HTTP call is made to either the email service or the script service, depending on the type of inject. These services receive the necessary information and then deploy the inject within the exercise infrastructure. For example, an email might be delivered to the relevant mail server, or a script could be executed on an attacker machine.

The first inject executed is the *"Web defacement preparation"* subprocess. When starting a subprocess, flowgin instantiates a new process, passes the required parameters (such as the attacker's IP and the target's IP), and starts it. The subprocess contains steps of a typical cyber kill chain, such as scanning, reconnaissance and placing a backdoor on the attacked server. Listing 1 shows a typical attack script, which is triggered by flowgin, by calling the script service, and then executed from the attacker against the participant infrastructure. The shown script is used for scanning and performs aggressive nmap scans. The attack script includes several "waiting times" to maintain realism. Just as in a manually executed attack by an attacker, there are natural pauses between commands. These waiting times simulate the delays that would occur during a real-life attack, adding to the authenticity of the exercise. Once all injects of the subprocess are completed, the main process continues. The next step is to use the backdoor placed on the web server (realized by the subprocess before) to replace the index page with a defaced page. This is achieved by flowgin again calling the script service, which receives the call and executes a script from the attacker that connects to the participant's web server (via the backdoor) and replaces the index page.

```
#!/bin/bash

echo "Loading configs/$1"
source configs/$1

echo "Executing nmap port discovery
    scan ..."

nmap -v -p- $TARGET

attack_wait 5s

echo "Executing service discovery on
    open ports ..."

nmap -v -sV -p $OPEN_PORTS $TARGET

echo "done."

attack_wait 5s

echo "Executing vhosts scan ..."

nmap -v --script http-vhosts --script-
    args "http-vhosts.domain=
    $TARGET_DOMAIN,http-vhosts.filelist
    =util/vhosts.lst" -p 80,443 $TARGET

attack_wait 10s

echo "Verifying vhosts ..."

for check in $CHECK_VHOSTS; do
```

```
echo "Checking $check ..."
curl -I --header "Host: $check"  https
    ://$TARGET -k
done

echo "Finished scans!"
```

Listing 1 Attack script of web scanning

After waiting for a duration of two minutes, the next inject, *"Check whether participant(s) detected the web defacement"* is executed. Therefore, the web server logs are queried to determine if a GET call has been made to the defaced index page. If a corresponding GET call is detected, the runtime variable "defacement" (which was initially set to "not detected") is set to "detected". Then, the runtime variable "defacement" serves as input for the following decision. If the defacement was detected, the process continues with a 5-minute wait period. If it was not detected, the inject *"Inform participant(s) about the defacement"* is executed beforehand. This inject technically sends a REST call to a web service on the email server, including the sender (with associated credentials for the email server), recipient, subject and content of an email. The email server's web service receives this information, logs onto the email service, and delivers the desired email to the recipient's mailbox. Afterwards, the 5-minute wait period is applied. Then, a check on the file system is performed to determine if the defacement or the backdoor still exist, and the results are accordingly stored in runtime variables.These runtime variables play a crucial role in the functioning of the process, as they act as global variables for the duration of a single scenario execution. In the current implementation, runtime variables are stored as strings, enabling them to capture a wide range of information, such as the presence or absence of specific files or conditions. When the process reaches a decision point, these runtime variables can be accessed and their values compared against predefined criteria to determine the appropriate routing for the workflow. For example, a runtime variable storing the status of the index file-whether defaced, replaced, or removed-can be checked, and the process can then branch accordingly to reflect the appropriate next steps. This approach allows for dynamic and data-driven decision-making during the execution of a scenario, ensuring that the process adapts in real-time based on the captured data. By maintaining global accessibility within the scenario execution, runtime variables enable seamless communication between tasks and enhance the flexibility and responsiveness of the workflow.

The entire flow, including the path decisions, is automated. The only manual aspect is gathering information from the infrastructure, such as determining defacement detection and checking for removal of the defacement or the backdoor. We opt for manual requests to showcase practicality while keeping the technical infrastructure as it was before. The results of these manual requests are passed to the process flow by adjusting runtime variables.

The utilization of a process engine allows for the automation of scenario execution, which (1) significantly reduces the complexity of conducting an exercise despite adaptive adjustments through various paths and (2) still maintains the necessary flexibility to respond to unexpected reactions of participants. During execution, event logs are collected, which represent a fundamental basis for the subsequent evaluating phase.

4.4 Evaluating

During the execution of the cyber exercise using flowgin, event logs are generated. For this purpose, a log event is written to a log file for each inject execution, containing the event name, timestamp, and the current values of runtime variables. Since each team has its own instantiation of the scenario in flowgin, these logs can be examined individually for each team. Such event logs can serve as input for process analysis tools such as process mining, allowing for both visualization and analysis of the processes followed by the teams. Beside of the potential use of process mining, observable runtime variables and associated event logs can serve as basis for calculating key performance indicators such as task detection / completion times, mean times per task, number of successful attacks, and more. Such indicators can be used as input, to extract evidence-based data from the learning environment [22] and can further be utilized to generate proper feedback. Therefore, runtime variables are carefully selected during the planning phase and filled during the conducting phase, where they also serve as decision indicators for choosing different paths. In the evaluation phase, they are potentially used for analysis, applying process mining algorithms, or calculating key performance indicators. Therefore, the proper selection of runtime variables is crucial to facilitate a smooth transition between the phases of the cyber exercise lifecycle.

```
instance-id: 100
instance-name: "Web Defacement Exercise
    Part"
...
events:
    - ...
    - event-name: "Check whether
       participant(s) detected the web
        defacement"
    event-timestamp: "2024-01-28T14
       :30:00"
    runtime-variables:
        - web-defacement: "detected"
        - backdoor: "not resolved"
  - ...
```

Listing 2 Event Log

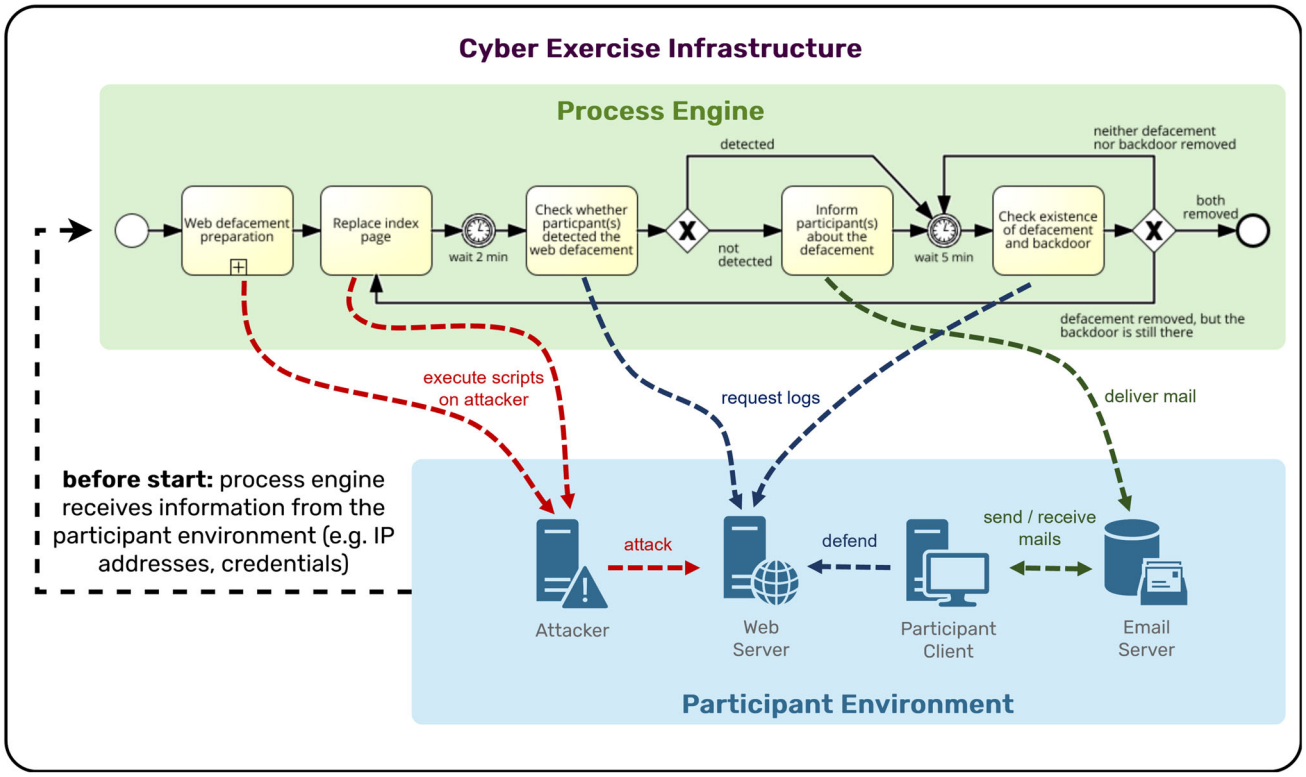

Fig. 6 Interplay between process engine and cyber exercise infrastructure (taken from [29])

Listing 2 shows an example event log in the form of a YAML [4] file. Lines 1-3 contain general information about the process that is executed. Starting from line 5, all events are displayed in the order in which they where executed during process enactment.

In context of our proof-of-concept, event logs were created, but process mining techniques such as process discovery [36] or conformance checking [31] were not applied due to the small number of teams involved. We will place a particular emphasis on creating use cases involving a larger number of teams in our future endeavors, to strengthen our argumentation with applicable examples for process mining.

By individually generating event logs, the behavior of participants in cyber exercises is (1) traceable and comparable with other exercise runs, and allows for (2) individual evaluation of participants or teams, resulting in more detailed feedback.

5 Case study

The proof-of-concept shown in Sect. 4 was applied in a representative case study during a large-scale national cyber exercise. The subsequent sections provide information about the cyber exercise itself, and the use case that was conducted.

5.1 National cyber exercise

The exercise involved four teams, each comprising 8-10 participants. The teams consisted of a well-balanced composition, with approximately half of the participants being technical experts and the remaining half holding managerial positions. The team members were essentially unfamiliar with each other, with the exception of Team D, where some participants had prior experience working together and demonstrated excellent coordination. In a fictitious scenario, each team represented a critical entity within a supply chain, spanning the manufacturing, transportation, retail, and IT service provider sectors. Consequently, each team managed its own respective infrastructure. Over a five-hour period, the exercise subjected the teams' infrastructures to various attack vectors, including website defacements, SQL injections, cross-site scripting, and misconfigurations.

The teams' objective was to effectively respond to these attacks through appropriate incident response management and communication, thereby defending against the assaults and preserving the integrity of the supply chain. Additionally, personnel from national Computer Emergency Response Teams (CERTs) and governmental institutions provided support and were present throughout the exercise.

Table 1 Event log analysis for teams

No.	Task	Team A	Team B	Team C	Team D
1	Web defacement preparation	09:42:38	09:43:25	11:27:23	13:39:49
2	Replace index page	09:44:22	09:45:59	11:28:48	13:40:38
3	Check whether participant(s) detected the web defacement	**09:46:22** defacement = detected	**09:47:59** defacement = detected	**11:30:48** defacement = detected	**13:42:38** defacement = not detected
4	Inform participants about the web defacement	-	-	-	**13:42:53** defacement = detected
5	Check existence of defacement and backdoor	**09:51:22** defacement = detected	**09:52:59** defacement = detected	**11:35:48** defacement = detected	**13:47:53** defacement = detected **13:52:53** defacement = removed
		09:56:22 defacement = removed	**09:57:59** defacement = removed	**11:50:48** defacement = removed	
6	Replace index page	-	-	-	**14:10:41** defacement = not detected
7	Check whether participant(s) detected the web defacement	-	-	-	**14:12:41** defacement = detected
8	Check existence of defacement and backdoor	-	-	-	**14:17:41** defacement = removed

5.2 Use case

The web defacement attack vector was used as a part of the extensive national cyber exercise to demonstrate our approach. The entire sequence of the case study can be traced based on the entries in Table 1. Since the exercise flow was slightly different for each team, the execution times of the web defacement attack varied, but they were performed in the same manner and on the same day. The different teams are labeled as A, B, C, and D for demonstration purposes.

Since each team had its own exercise infrastructure and different attacker machines existed, the following information was provided to the process (in the form of variables) before its instantiation:

- Attacker's IP address (240.172.53.0; The same attacker machine was used for all teams. If a team would have already blocked this attacker, the process could be restarted with a different attacker IP).
- Web server domain (www.A-D.com)
- Email server IP (10.0.1-4.100)

These IP addresses and domains are fictitious elements that exist only within the exercise infrastructure. Additional credentials for the email server did not need to be provided because a master access was implemented, which is universally valid. Therefore, this information does not need to be dynamically provided at the process start but can be statically integrated into the body of the corresponding inject (i.e., "Inform participants about web defacement") beforehand.

After all the necessary variables were passed to the process, it was started. Table 1 shows the injects with associated timestamps and the development of the runtime variable *defacement* (which can take three different values: *not detected*, *detected* and *removed*). In line one of Table 1, the execution time of the first inject (i.e., subprocess "Web defacement preparation") is depicted. After 1-3 min (depending on how long the subprocess took), the preparation for the web defacement was successfully completed, and the next inject "Replace index page" was executed. After a two-minute wait (as foreseen in the process model), it was checked whether the participants had already detected the defacement. Therefore, a person from the organising team checked in the web server logs if after the defacement there has already been a GET call to the web server. In our case study, three out of the four teams had already detected the defacement. Only Team D had not yet discovered it, which is why they were the only team that received an information email about the defacement (see line number 4). By sending out this information email, the runtime variable was set to *detected*.

After a 5-minute wait (as foreseen in the process model), a check was made to see if the teams had already removed

the defacement or the backdoor. Since no team discovered and removed the backdoor throughout the entire exercise, we will not place further emphasis on the backdoor in this use case. As shown in line 5, no team was able to remove the defacement within the first 5 min. Teams A, B, and D were able to remove it after 10 min. Team C removed it after a total of 20 min.

For the occurred case in which no team removed the backdoor, the original plan of the process entailed was a repetition of the defacement. Nevertheless, given the considerable challenges faced by the teams owing to prior attacks, a collective decision was made to refrain from executing another defacement for Teams A, B and C. However, it was evident that Team D remained relatively unchallenged, so we decided to resume their process at 2:10:41 PM. Consequently, the index page was subjected to defacement once again, and the runtime variable *defacement* was set back to *not detected*. This time, the team promptly detected the defacement and was able to remove it within the initial five minutes. Nonetheless, the backdoor was still present.

In accordance with our approach, we employed the gathered event logs to conduct objective assessments and offer appropriate feedback to the participants. Accordingly, we have determined, that in essence, all teams exhibited very similar performance levels during the exercise, with comparable response times to detect and resolve the attacks. Teams A, B, and D notably succeeded in removing the website defacement within 10 min of detection, while Team C required 20 min to accomplish the same task. However, none of the teams were able to identify and eliminate the backdoor vulnerability. Therefore, given the overarching context of the exercise, which also encompassed other attack vectors beyond the scope of this specific use case, it became evident that Team D, in particular, delivered an exceptional performance. This achievement may be attributed to preexisting familiarity and professional collaboration among some team members, while the other teams were randomly compiled with people from different organizations. Consequently, Team D was the only team for which we reintroduced the website defacement to demand its participants with additional challenges and prevent them from being underchallenged.

Planning the use case in a process model incorporating various paths and decision variables in combination with executing it in a process engine significantly improved the efficiency of the cyber exercise. Our case study demonstrated that, compared to other parts of the exercise where our approach was not applied, this segment required significantly less effort during the exercise itself, despite being far more dynamic. Furthermore, we were able to produce detailed evaluation results without relying on observers, as the process engine automatically logged relevant data points for analysis. This reduction in required personnel and manual effort highlights the increased efficiency enabled by our approach. Moreover, our dynamic approach ensured that teams facing minimal challenges were demanded with additional tasks, while those grappling with overwhelming obstacles were either provided with helpful guidance or spared from additional burdens. The participants of Team D, in particular, serve as a compelling example that demonstrates how they might have been under-challenged in a linearly structured cyber exercise but got an additional task assigned thanks to our flexible approach, ensuring they were appropriately challenged. Therefore, we created a customizable exercise environment while ensuring a high degree of automation to accommodate rapid adaptations.

6 Discussion

The process-based approach to planning, conducting, and evaluating cyber exercises opens up a wide range of future possibilities for enhancing cybersecurity training. By leveraging standardized process modeling languages, process engines, and process evaluation methods / metrics, this approach offers a structured, scalable, and adaptable framework for developing comprehensive cyber exercises. In this section, we discuss potential applications of this methodology, its implications for the future of cybersecurity training, and its limitations.

6.1 Cybersecurity interactive training environments

The process-based approach to cyber exercises has the potential to revolutionize cybersecurity training by enabling a highly interactive, self-directed learning experience. In this model, participants can independently initiate a process that runs through a predefined scenario, allowing them to engage with the exercise at their own pace and according to their individual learning needs. By leveraging process engines, the scenario is automatically executed, with each step of the process carefully tracked and logged. This tracking mechanism captures every action taken by the participant, providing a detailed record of their progress and performance throughout the exercise.

What makes this approach particularly powerful is the ability to automate scenario-based decision-making and dynamically adjust the scenario based on the participant's actions. For example, if a participant successfully mitigates a web server attack by correctly reconfiguring the server, this action can be automatically detected through the logs generated and requested by the process engine. The system can then trigger a new, more advanced attack as a direct response to the participant's success, creating a continuous and evolving challenge that adapts in real-time. This level of interactivity transforms the exercise into a dynamic expe-

rience similar to a computer game, where the environment responds to the player's actions, creating a sense of immersion and engagement.

Such a training environment offers several benefits for cybersecurity education. First, it allows participants to learn by doing, engaging in practical problem-solving and decision-making in a simulated but realistic context. The ability to interact with the scenario and see the immediate consequences of their actions helps reinforce learning and build critical skills. Second, because the process is automated and self-directed, participants can repeat exercises as needed to refine their skills or try different strategies, fostering a deeper understanding of the material and promoting a more personalized learning experience.

Moreover, the use of automated tracking and adaptive scenarios reduces the need for constant oversight from instructors, making it possible to scale the training to accommodate more participants or more complex exercises without a corresponding increase in resources. This scalability is further enhanced by the ability to have multiple teams participate in the same or different scenarios concurrently, with minimal overhead. This is particularly valuable in a field like cybersecurity, where the need for skilled professionals is constantly growing, and traditional training methods may struggle to keep pace with demand.

The process-based approach also enables the creation of a structured learning path, where each scenario builds on the previous one, gradually increasing in complexity as the participant's skills develop. This progression mirrors the learning curve of a computer game, where early levels introduce basic concepts and skills, and later levels present increasingly challenging scenarios that require mastery of those skills. By structuring training in this way, organizations can ensure that participants are not only learning but also applying their knowledge in progressively more difficult and realistic situations, preparing them for the complex and dynamic nature of real-world cybersecurity threats.

In general, the process-based approach offers an innovative way to deliver interactive and self-directed cybersecurity training. By allowing participants to independently engage with scenarios, automatically tracking their progress, and dynamically adapting to their actions, this approach creates a highly engaging and effective learning environment. It combines the best elements of hands-on training and computer-based learning, offering a scalable and flexible solution to the growing demand for cybersecurity education. As the field continues to evolve, the ability to provide responsive, adaptive training will be crucial for preparing the next generation of cybersecurity professionals to meet the challenges of an increasingly digital world.

6.2 Automated feedback mechanism

The process-based approach to cybersecurity exercises offers significant potential for incorporating automated feedback mechanisms, which can greatly enhance the learning experience for participants. By leveraging the detailed logs and data captured by process engines during exercises, the system can provide immediate and continuous feedback on participant actions and decisions. For example, if a participant successfully mitigates a simulated attack or correctly implements a security configuration, the process engine can automatically acknowledge this success and offer constructive feedback, highlighting what was done correctly and why it was effective. Conversely, if a participant makes an error or fails to respond adequately to an inject, the system can provide timely guidance or suggest alternative strategies, helping the participant understand what went wrong and how to improve.

In addition to immediate feedback, the data captured by the process engine can be used to generate detailed metrics, such as response times, task completion rates, and decision-making patterns. These metrics can play a vital role in the creation of After Action Reports (AARs), offering a structured and data-driven analysis of exercise performance. By automating the generation of such metrics, the system reduces the manual workload for instructors and evaluators while ensuring a consistent and objective basis for post-exercise evaluations. These metrics not only inform participants of their performance but also provide valuable insights for improving future exercises.

These automated feedback mechanisms transform cybersecurity training into an interactive and responsive learning environment. Participants are able to learn from their mistakes in real time and adjust their approach as needed, which promotes a deeper understanding of the material and accelerates skill development. Moreover, automated feedback reduces the need for constant instructor oversight, allowing training programs to scale up more easily and accommodate more participants without sacrificing the quality of instruction. By providing instant, tailored feedback based on individual performance, this approach supports a personalized learning experience, catering to the unique needs of each participant and enhancing overall engagement and retention.

6.3 Multi-scenario coordination across cyber ranges

The process-based approach to cybersecurity exercises enables the coordination of multiple scenarios across different cyber ranges, allowing for more comprehensive and realistic training experiences. By utilizing a centralized process engine, various scenarios can be managed concurrently, even if they are running on separate infrastructures. This capability is particularly valuable for simulating complex, multi-faceted cyber threats that require collaboration and

interaction across different teams and environments. To facilitate this multi-scenario coordination, endpoints must be provided to interface with each cyber range, allowing the process engine to deploy injects and control the flow of each scenario seamlessly.

These endpoints serve as connection points, ensuring that injects and commands are delivered accurately and on time, regardless of the underlying infrastructure. This setup allows for the integration of diverse training environments, creating a unified exercise experience that can encompass different network topologies, systems, and organizational structures. The ability to manage multiple scenarios across various cyber ranges not only enhances the realism and depth of the exercises but also enables organizations to conduct large-scale, distributed training sessions that mirror real-world situations. By providing this level of flexibility and scalability, the process-based approach allows for connectivity between multiple cyber ranges and, therefore, supports a more dynamic and robust training environment, better preparing participants for the complexities of modern cybersecurity challenges.

6.4 Limitations

While the process-based approach to plan, execute and evaluate cyber exercises offers numerous advantages in terms of scalability, automation, and adaptability, it is not without its limitations. One of the primary challenges lies in the significant upfront complexity and effort required during the planning phase. Designing comprehensive scenarios involves identifying meaningful decision pathways, perform detailed process modeling, defining adaptive triggers, discovering appropriate log files for evaluation, and establishing appropriate endpoints for seamless communication between the process engine and the underlying infrastructure. This initial configuration demands technical expertise and careful planning to ensure that all components - from the process engine to the supporting infrastructure - function cohesively. However, this initial complexity is a logical consequence of creating far more dynamic, flexible, and complex cyber exercise scenarios compared to static ones. Planning a dynamic scenario with multiple pathways inherently requires more time and effort than a simple static exercise, but it also significantly enhances the quality of the training experience. Moreover, in order to mitigate the limitations mentioned, our approach allows for a modular design. This modularity allows individual components of the scenario, such as injects, decision points, and evaluation criteria, to be designed, tested, and refined independently before being integrated into the larger exercise workflow. By isolating these components during the implementation phase, the complexity of the overall setup is reduced, and planners can focus on ensuring the quality and functionality of each part before combining them. This modular approach not only simplifies the configuration process but also makes it easier to adapt or extend specific elements in response to changes in objectives or requirements, ensuring a flexible yet efficient setup process. However, once this initial effort is completed, subsequent exercises benefit from reusability and repeatability, reducing the workload for the planning of future scenarios. The high level of automation further minimizes effort during the exercises themselves, making the approach more efficient over time.

7 Conclusion and future work

We have developed a process-based lifecycle model for planning, conducting, and evaluating customized cyber exercises. We have used the acknowledged ENISA cyber exercise lifecycle and extended it with methods from Business Process Management in order to improve the effectiveness of cyber exercises as well as increase their level of automation. For this purpose, we utilized the process modeling language BPMN to plan an adaptive cyber exercise scenario, whose model is then executed in a process engine. During conducting the exercise, comprehensive event logs, including runtime variables, are documented to provide a solid foundation for gaining process insights in the evaluating phase. Additionally, our approach is holistic and enables seamless transitions between phases of the lifecycle. The process model from the planning phase is directly executed by a process engine during the conducting phase, and the logs from the conducting phase are then used to gain detailed information on participant behaviour and exercise progress in the evaluating phase. We explicitly presented this approach as a concept and demonstrated its feasibility through a proof-of-concept implementation. This was further applied within a case study as part of a national exercise, which successfully validated the practical applicability of the approach in a realistic setting.

Two significant improvements were identified through this application:

1. *Increase the quality of cyber exercises:* Our approach enables the development of more complex exercises that include different paths to adapt flexibly to participants' skill levels. Furthermore, the individual assessment of teams and their chosen paths through a scenario allows for more targeted feedback.
2. *Increase the efficiency and effectiveness of conducting cyber exercises.* The clear representation of cyber exercises in a process model allows for a more straightforward depiction of complex scenarios. Moreover, the use of subprocesses enhances the reusability of exercise components, and the utilization of process engines significantly increases automation and thus efficiency.

Moreover, our comprehensive discussion highlights the significant potential that this process-based approach brings for future cyber exercises. By leveraging process modeling and automation, this approach not only enhances the complexity and adaptability of training scenarios but also enables the integration of multi-scenario coordination across different cyber ranges. This flexibility allows for the creation of dynamic, self-directed learning environments where participants can engage with tailored exercises that adjust in real-time based on their actions. Additionally, the use of process engines and automated feedback mechanisms can streamline the management of large-scale exercises, providing a scalable solution for training diverse teams across various infrastructures. These advancements pave the way for more interactive, immersive, and efficient cyber training experiences, setting a new standard for how cyber exercises are designed and executed.

For future work, we plan to expand our approach in all phases of the ENISA lifecycle to conduct increasingly extensive and more complex cyber exercises. Our aim is to continuously enhance the exercise experience for participants while simultaneously increasing the level of automation, thereby maximizing the potential for this approach to be adopted widely as a feasible solution for skill development. In our future work, we plan to place special focus on the areas highlighted in the "Discussion" chapter, such as developing interactive cybersecurity training environments, enhancing automated feedback mechanisms, and enabling multi-scenario coordination across cyber ranges. Additionally, we intend to leverage the detailed logs generated by the scenario-executing process engine to perform process mining, which will help us extract insights into scenario execution, identify inefficiencies, and further optimize the design and flow of future exercises. By exploring these aspects further, we aim to push the boundaries of what cyber exercises can achieve, providing more immersive, adaptive, and comprehensive training experiences. These efforts will contribute to creating a more effective and scalable framework for cybersecurity education and preparedness.

Funding Open access funding provided by AIT Austrian Institute of Technology GmbH. The research leading to these results received funding from *The Austrian Research Promotion Agency (FFG)* under the Grant Agreement Number *FO999888556*.

Declarations

Financial interests The authors have received research support from their respective affiliations.

Non-financial interests The authors have no relevant non-financial interests to disclose.

References

1. Andreolini, M., Colacino, V.G., Colajanni, M., Marchetti, M.: A framework for the evaluation of trainee performance in cyber range exercises. Mob. Netw. Appl. **25**(1), 236–247 (2020)
2. Angafor, G.N., Yevseyeva, I., He, Y.: Game-based learning: a review of tabletop exercises for cybersecurity incident response training. Secur. Priv. **3**(6), e126 (2020)
3. Arcus, R., Christoforatos, N., Fanourakis, F., Fernández, G., Van Heurck, C., Zacharis, A.: Cyber Europe 2024: After Action Report. (2024)
4. Ben-Kiki, O., Evans, C., Ingerson, B.: (2009) Yaml ain't Markup Language (yaml^{TM}) version 1.1. Working Draft 2008 5(11)
5. Birkland, T.A.: Disasters, lessons learned, and fantasy documents. J. Conting. Crisis Manag. **17**(3), 146–156 (2009)
6. Brilingaite, A., Bukauskas, L., Juozapavicius, A.: A framework for competence development and assessment in hybrid cybersecurity exercises. Comput. Secur. **88**, 101607 (2020)
7. Čeleda, P., Čegan, J., Vykopal, J., Tovarňák, D., et al.: Kypo-a platform for cyber defence exercises. M&S Support to Operational Tasks Including War Gaming, Logistics, Cyber Defence NATO Science and Technology Organization (2015)
8. Christoforatos, N., Lella, I., Rekleitis, E., Van Heurck, C., Zacharis, A.: Cyber Europe 2022: After Action Report (2022)
9. Delima, R., Wardoyo, R., Mustofa, K.: Goal-oriented requirements engineering: state of the art and research trend. JUITA J. Inform. **9**(1), 105–114 (2021)
10. Doupé A, Egele M, Caillat B, Stringhini G, Yakin G, Zand A, Cavedon L, Vigna G.: Hit 'em where it hurts: a live security exercise on cyber situational awareness. In: Proceedings of the 27th Annual Computer Security Applications Conference, ACM, ACSAC '11, pp. 51–61, https://doi.org/10.1145/2076732.2076740(2011)
11. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A., et al.: Fundamentals of Business Process Management. Springer, New York (2018)
12. ENISA - European Network and Information Security Agency.: Good Practice Guide on National Exercises. https://www.enisa.europa.eu/publications/national-exercise-good-practice-guide (2009)
13. Furtună, A., Patriciu, VV., Bica, I.: A structured approach for implementing cyber security exercises. In: 2010 8th International Conference on Communications, IEEE, pp 415–418 (2010)
14. Jans, M.J., Alles, M., Vasarhelyi, M.A.: Process Mining of Event Logs in Auditing: Opportunities and Challenges. Available at SSRN 1578912 (2010)
15. Karjalainen, M., Kokkonen, T., Puuska, S.: Pedagogical aspects of cyber security exercises. In: 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), IEEE, pp 103–108 (2019)

16. Karjalainen, M., Puuska, S., Kokkonen, T.: Measuring learning in a cyber security exercise. In: Proceedings of the 12th International Conference on Education Technology and Computers, pp 205–209 (2020)
17. Keller, G., Scheer, AW., Nüttgens, M.: Semantische Prozeßmodellierung auf der Grundlage" Ereignisgesteuerter Prozeßketten (EPK)". Inst. für Wirtschaftsinformatik (1992)
18. 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)
19. Laplante, P.A., Kassab, M.: Requirements Engineering for Software and Systems. Auerbach Publications, Boca Raton (2022)
20. Leitner, M.: A scenario-driven cyber security awareness exercise utilizing dynamic polling: Methodology and lessons learned. In: Proceedings of the 9th International Conference on Information Systems Security and Privacy - ICISSP, INSTICC, SDciTePress, pp. 634–64. https://doi.org/10.5220/0011780400003405 (2023)
21. Leitner, M., Frank, M., Hotwagner, W., Langner, G., Maurhart, O., Pahi, T., Reuter, L., Skopik, F., Smith, P., Warum, M.: Ait cyber range: flexible cyber security environment for exercises, training and research. In: Proceedings of the European Interdisciplinary Cybersecurity Conference, pp. 1–6 (2020)
22. Maennel, K.: Learning analytics perspective: Evidencing learning from digital datasets in cybersecurity exercises. In: 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), IEEE, pp. 27–36 (2020)
23. Manger, J., Rinderle-Ma, S.: Cloud Process Execution Engine: Architecture and Interfaces. arXiv preprint arXiv:2208.12214 (2022)
24. Mangler, J., Stuermer, G., Schikuta, E.: Cloud Process Execution Engine-Evaluation of the Core Concepts. arXiv preprint arXiv:1003.3330 (2010)
25. Mäses, S., Maennel, K., Toussaint, M., Rosa, V.: Success factors for designing a cybersecurity exercise on the example of incident response. In: 2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), IEEE, pp. 259–268 (2021)
26. Mendling, J., Reijers, H.A., van der Aalst, W.M.: Seven process modeling guidelines (7pmg). Inf. Softw. Technol. **52**(2), 127–136 (2010)
27. OMG.: Business Process Model and Notation (BPMN), Version 2.0. Object Management Group, http://www.omg.org/spec/BPMN/2.0 (2011)
28. Petersen, R., Santos, D., Smith, M., Witte, G.: Workforce Framework for Cybersecurity (Nice Framework). Tech. rep, National Institute of Standards and Technology (2020)
29. Pfaller, T., Skopik, F., Smith, P., Leitner, M.: Towards customized cyber exercises using a process-based lifecycle model. In: European Interdisciplinary Cybersecurity Conference, pp. 37–45 (2024)
30. Rajasekharaiah, K., Dule, CS., Sudarshan, E.: Cyber security challenges and its emerging trends on latest technologies. In: IOP Conference Series: Materials Science and Engineering, IOP Publishing, vol. 981, p. 022062 (2020)
31. Rozinat, A., Van der Aalst, W.M.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. **33**(1), 64–95 (2008)
32. Seker, E., Ozbenli, HH.: The concept of cyber defence exercises (cdx): Planning, execution, evaluation. In: 2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), IEEE, pp. 1–9 (2018)
33. Shin S, Seto Y (2020) Development of iot security exercise contents for cyber security exercise system. In: 2020 13th International Conference on Human System Interaction (HSI), IEEE, pp 1–6
34. Skopik, F., Leitner, M.: Preparing for national cyber crises using non-linear cyber exercises. In: 2021 18th International Conference on Privacy, pp. 1–5. IEEE, Security and Trust (PST) (2021)
35. van Der Aalst, W.M., Ter Hofstede, A.H., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distrib. Parallel Databases **14**, 5–51 (2003)
36. van der Aalst, W.M.: Process discovery: capturing the invisible. IEEE Comput. Intell. Mag. **5**(1), 28–41 (2010)
37. Van Der Aalst, W.: Process mining. Commun. ACM **55**(8), 76–83 (2012)
38. Virág, C., Čegan, J., Lieskovan, T., Merialdo, M.: The current state of the art and future of european cyber range ecosystem. In: 2021 IEEE International Conference on Cyber Security and Resilience (CSR), IEEE, pp. 390–395 (2021)
39. Vykopal, J., Vizvary, M., Oslejsek, R., Celeda, P., Tovarnak, D.: Lessons learned from complex hands-on defence exercises in a cyber range. In: 2017 IEEE Frontiers in Education Conference (FIE), p. 1. https://doi.org/10.1109/FIE.2017.8190713 (2017)
40. Vykopal, J., Ošlejšek, R., Burská, K., Zákopčanová, K.: Timely feedback in unstructured cybersecurity exercises. In: Proceedings of the 49th ACM Technical Symposium on Computer Science Education, pp. 173–178 (2018)
41. Weiss, R., Locasto, ME., Mache, J.: A reflective approach to assessing student performance in cybersecurity exercises. In: Proceedings of the 47th ACM Technical Symposium on Computing Science Education, pp. 597–602 (2016)
42. Wen, S.F., Yamin, MM., Katt, B.: Ontology-based scenario modeling for cyber security exercise. In: 2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), IEEE, pp. 249–258 (2021)
43. Weske M, et al.: Concepts, languages, architectures. Business Process Management. (2007)
44. Wright, C.V., Mache, J., Weiss, R.: Hands-on exercises about DNS attacks: details, setup and lessons learned. J. Comput. Sci. Coll. **32**(1), 117–125 (2016)
45. Yadav, T., Rao, AM.: Technical aspects of cyber kill chain. In: Security in Computing and Communications: Third International Symposium, SSCC 2015, Kochi, India, August 10-13, 2015. Proceedings 3, Springer, pp. 438–452 (2015)
46. Yamin, MM., Katt, B.: Inefficiencies in cyber-security exercises life-cycle: A position paper. In: AAAI Fall Symposium: ALEC, pp. 41–43 (2018)
47. Yamin, M.M., Katt, B.: Modeling and executing cyber security exercise scenarios in cyber ranges. Comput. Secur. **116**, 102635 (2022)