Managing cyber defence training scenarios with Skade

Automating threat hunting challenges

TEODOR SOMMESTAD, LOVISA NYHOLM,
HENRIK KARLZÉN, HANNA KVIST

FOI-R--5968--SE
ISSN 1650-1942 September 2026

Teodor Sommestad, Lovisa Nyholm, Henrik Karlzén, Hanna Kvist

Managing cyber defence training scenarios with Skade

Automating threat hunting challenges

Titel	Hantering av träningsscenarier för cyberförsvar med Skade – Automatisering av hotjaktsutmaningar
Title	Managing cyber defence training scenarios with Skade – Automating threat hunting challenges
Rapportnr/Report no	FOI-R--5968--SE
Månad/Month	September
Utgivningsår/Year	2026
Antal sidor/Pages	26
ISSN	1650-1942
Uppdragsgivare/Client	Myndigheten för civilt försvar
Forskningsområde	Cyberförsvar och cybersäkerhet
FoT-område	Inget FoT-område
Projektnr/Project no	B73015
Godkänd av/Approved by	Foteini Papiri
Ansvarig avdelning	Cyberförsvar och ledningsteknik

Detta verk är skyddat enligt lagen (1960:729) om upphovsrätt till litterära och konstnärliga verk, vilket bl.a. innebär att citering är tillåten i enlighet med vad som anges i 22 § i nämnd lag. För att använda verket på ett sätt som inte medges direkt av svensk lag krävs särskild överenskommelse.

This work is protected by the Swedish Act on Copyright in Literary and Artistic Works (1960:729). Citation is permitted in accordance with article 22 in said act. Any form of use that goes beyond what is permitted by Swedish copyright law, requires the written permission of FOI.

Sammanfattning

Hotjakt möjliggör ett proaktivt försvar mot avancerade cyberattacker och är en kompetens som eftersöks i många organisationer. Samtidigt är hotjakt svårt och kräver träning. FOI har utvecklat och utvärderat webbapplikationen Skade, som är utformad för att ge deltagare praktiska och tekniska utbildningsscenarier utan att en mänsklig instruktör behöver vara inblandad. Skade bygger på en scenariospecifikation och några funktioner som är vanliga i plattformar som används för att producera tekniska utbildningsscenarier, exempelvis verktyg för hotemulering. Applikationen testades med 45 deltagare och fick ett godtagbart resultat (67 %) enligt System Usability Scale (SUS).

Skade har utformats för att balansera svårighetsgraden genom att erbjuda ledtrådar som deltagaren kan använda om de kör fast. Våra tester visade dock att denna funktion inte gjorde någon skillnad i varken upplevt lärande eller upplevd svårighetsnivå. Verktyg som Skade kan alltså automatisera genomförandet av praktisk utbildning i cyber ranges, men projektet har inte demonstrerat några enkla sätt att anpassa svårighetsgraden i uppgifterna.

Nyckelord: cyber range, hotjakt, träning, utbildning

Summary

Cyber threat hunting offers a proactive defence to advanced cyber attacks, and is a skill wanted in many organizations. However cyber threat hunting is difficult, and requires training. FOI has developed and tested the web application Skade, which is designed to expose technical hands-on training scenarios in cyber ranges to trainees, without the involvement of a human instructor. Skade rests on a scenario specification language and some common functions of cyber ranges, e.g., threat emulation tools. It was tested with 45 trainees, who gave it an acceptable assessment (67%) on the System Usability Scale.

Skade was designed to balance difficulty level with clues that the trainee could use if they got stuck. The tests showed that this feature was ineffective, and made no difference in perceived learning or perceived difficulty level. Thus, while tools like Skade can automate the execution of hands-on training in cyber ranges, the project did not demonstrate a solution for the issue of balancing difficulty level, and guide trainees.

Keywords: cyber range, threat hunting, training, education

Table of Contents

1 Introduction 6
1.1 Cyber threat hunting 6
1.2 Constructing threat hunting training 7
1.3 Project objectives 9
2 The Skade application 10
2.1 Design principles 10
2.1.1 Multiple learning dimensions 10
2.1.2 Constructive alignment 11
2.1.3 Feedback and assessment 11
2.1.4 Motivating setting 12
2.2 The scenario specification language 12
2.3 Application design and integrations 16
3 Field tests 18
3.1 Test design 18
3.2 Overall system usability 19
3.3 Effect of clues 20
4 Discussion and future work 22
5 References 24
Appendix: Learning objectives 26

1 Introduction

This report summarizes the results of a research project funded by the Swedish Civil Contingencies Agency, later the Swedish Civil Defence and Resilience Agency, during the period 2021–2026. The project was a part of the collaboration effort between the USA and Sweden, under Technical Annex 07, pursuant to the Project Arrangement 016-2011. This report is therefore written in English.

The project focused on techniques for producing educational challenges for those aiming to learn *cyber threat hunting*. In this chapter we give an introduction to the area and project by a general introduction to cyber threat hunting and hands on training. We also give an overview of the objectives of the Skade project.

1.1 Cyber threat hunting

Cyber threat hunting has been described as “the human-driven, proactive and iterative search through networks, endpoints, or datasets in order to detect malicious, suspicious, or risky activities that have evaded detection by existing automated tools” (sqrrl, 2018). The traces of these malicious, suspicious, and risky activities are sometimes called indicators of compromise (*IoCs*), i.e. signs that systems have been compromised. The IoCs are typically seen as clues a threat hunter needs to piece together to make sure that an observed activity is truly malicious.

A threat hunt tends to start because new threat intelligence has become available or because something strange appears to have happened in the computer network. However, it could also be initiated just as a precaution, e.g., because someone is a bit worried about the security posture of a network. From this starting position, a threat hunter forms a concrete hypothesis about a threat. For example, when a log analyst has seen a recent increase in traffic using port 1717, and a threat hunter could hypothesize: *I believe that an attacker is exfiltrating data from our network through port 1717.*

In general, threat hunting is concerned with finding threats that have evaded detection by existing automated tools such as intrusion detection systems. Thus, a threat hunter does things that are similar to what other cyber security analysts do, but the threat hunter typically looks at things that are difficult to enumerate beforehand. More specifically, if the IoCs were easy to enumerate beforehand, they would probably have been handled by the existing automated tools, and no threat hunt would be needed. Consequently, it has been said that “threat hunting is a highly unstructured task that demands deep technical know-how, data analytics savvy and out of the box thinking” (Miazi et al., 2017).

Thus, a threat hunter often needs to solve a puzzle of many indicators and possible explanations to try to reject the hypothesis. These pieces include:

- system logs, e.g., access logs or network connections
- system states, e.g., the runtime memory or files on machines
- threat intelligence, e.g., IP addresses that are known to be malicious
- general cyber security knowledge, e.g., facts about access controls
- situated knowledge, e.g., what is normal in the organization

The knowledge, skills, and abilities required by a threat hunter are similar to many other cyber security roles. In relation to the specialties listed in the NICE framework for the cyber workforce (Petersen et al., 2020), threat hunting falls somewhere in between specialties such as *Defensive Cybersecurity*, *Threat Analysis*, and *Incident Response*. Overlaps with other specialties are helpful as individuals who aspire to be threat hunters can train relevant skills in courses developed for other, more common, cyber security specialties. However, as Miazi et al. (Miazi et al., 2017) express, the activities carried out during threat hunting are "highly unstructured", and threat hunting requires skills that need to be combined in different ways. For instance, threat hunting may require reasoning about potential unknown threats or writing code to correlate logs in novel ways.

Despite threat hunting not being a separate role or specialty recognized in NIST's workforce framework for cybersecurity, threat hunting expertise is in demand in the job market. This is, among other things, evident from training opportunities made available to practitioners in this field. For example, training in the process of threat hunting, and tasks clearly associated with threat hunting, are offered in courses aimed at professionals by organizations such as SANS in its course FOR508[1], and in training platforms such as Cybrary [2].

1.2 Constructing threat hunting training

The need for training and education on threat hunting is, as for many other cyber security specialties, substantial. Because threat hunting is a practical puzzle solving activity, a natural way to approach learning is to create practical puzzles and guide a trainee to solve these puzzles. This requires access to:

- A relevant computer network, e.g., with the type of systems and logs that a threat hunter would see in real operational networks.
- Cyber security events with a ground truth, e.g., legitimate user activities and attacks together with a ground truth saying who did what.

[1] https://www.sans.org/cyber-security-courses/advanced-incident-response-threat-hunting-training
[2] https://www.cybrary.it

There is a plethora of commercial course offerings and certifications for threat hunters. These typically involve practical activities where the trainee is to identify a threat with various tools. For instance, the SANS course "FOR508: Advanced Incident Response, Threat Hunting, and Digital Forensics" covers 35 practical labs. The courses target threat hunting. However, their labs are static, and cannot be easily modified to create new training material.

In addition, the labs and exercises tend to focus on isolated technical problems. Threat hunting, however, is not simply the use of predefined tools and technical skills, but also an investigative process. Working in this process requires comprehensive and connected training scenarios, e.g., integrating threat intelligence with logs and user behaviour. Complex scenarios integrating many steps of the process are available in large cyber defence exercises. For instance, CISA's President's Cup[3] involves complex threat hunting scenarios, and NATO CCDCOE's exercise Crossed Swords[4] involves threat hunting challenges. However, unlike the practical labs, these larger exercises are one-off, have many loose ends, and require human supervision in terms of exercise leads etc. Thus, generating complex and coherent threat hunting challenges automatically would be of great benefit. If the exercises could be executed without human supervision it is even better.

There are publicly available tools that are built to reduce the amount of human supervision and work in relation to exercises. For instance, the Software Engineering Institute at Carnegie Mellon University has produced tools such as:

- The Cyber Decision Analysis Simulator (CDAS)[5], which generates cyberattack scenarios for use in cyber training.
- The TopoMojo[6] tool, which simplifies the creation and deployment of virtual machine training environments.
- STEPfwd platform[7], which contains a library of instructions, reference information, and hands-on labs.

These toolsets could theoretically be used to produce threat hunting challenges, and manage the instantiation of challenges dynamically. Thus, if scenario generators such as CDAS are connected to networks, event simulators, and more, the scenario generation for threat hunting could be automated. However, this is far from trivial. This is the gap targeted in the research presented in this report: to create a technical solution that can generate and manage threat hunting challenges automatically.

[3] https://www.cisa.gov/topics/cybersecurity-best-practices/presidents-cup-cybersecurity-competition
[4] https://ccdcoe.org/crossed-swords/
[5] https://github.com/cmu-sei/CDAS
[6] https://github.com/cmu-sei/TopoMojo
[7] https://step-web.heinz.cmu.edu/lms

A known problem in the generation of educational cyber security challenges is the management of the difficulty level of the challenges. In hands-on labs there is often an instructor available to help, and in larger exercises there are exercise leads controlling the tempo. Calibrating this level is crucial for automated management of cyber security challenges, and it is widely recognized as a difficult problem. In the academic literature, it is clear that several researchers struggle with balancing the difficulty level of challenges to meet the proficiency level of trainees. For instance, Pfaller et al. (2024) suggest that exercises should be flexible and adapted to each team or participant. Various approaches have been taken to handle this. For example, Beuran et al. (2019) offer optional hints in their challenges and Seda et al. (2021) try to adapt upcoming challenges based on performance on previous ones. Commercial platforms also work in adapting the difficulty level. For instance, CyberBit (2020) offers an option to “allow hints on quiz” during exercise planning. This issue of managing difficulty levels was identified as the main obstacle of the automation effort.

1.3 Project objectives

The aims of the Skade project largely overlapped with the above-mentioned efforts. More specifically, the Skade project aimed to integrate a learning management system with cyber range tools. The project aims to:

- Create and develop a threat hunting scenario specification language. The language should be used to describe the ground truth, and how to instrument cyber range systems.
- Develop a prototype called Skade, capable of instantiating different threat hunting training scenarios with different properties (e.g., difficulty levels) in a cyber range.
- Evaluate Skade and the training cases it generates. Evaluations would cover both how well the difficulty settings match the perceived difficulty of trainees, and how much trainees learn from the scenarios.

Chapter 2 of this report describes the Skade prototype, including its design principles, the scenario specification language and the technical architecture of the prototype. Chapter 3 describes the evaluation with field tests and their results. Chapter 4 discusses the results and the future outlook for fully automated threat hunting challenges.

2 The Skade application

The sections below describe the main design principles of the Skade web application, the technical scenario specification language, and how the web application interacts with various cyber range support tools to produce the technical challenges.

2.1 Design principles

The design principles of Skade are laid out and motivated in the research paper written by Sommestad et al. (2023). As stated in that paper, there are four main ideas behind Skade:

- covering multiple learning dimensions
- ensuring constructive alignment
- providing feedback and assessment
- supporting a motivating setting

These ideas are often applied in hands-on training of various forms. They are further explained below.

2.1.1 Multiple learning dimensions

Most hands-on training covers multiple learning dimensions. That is, they align with Kolb's (1984) idea of offering a loop of abstract conceptualization, active experimentation, concrete experience, and reflective observation. This learning loop fits well with typical descriptions of threat hunting, as Figure 1 shows. Thus, the overall premise of hands-on threat hunting training is aligned well with this established pedagogical model requiring that the trainee should be active. A design principle in Skade was to ensure that each session covered all steps, e.g., by posing questions that encourage reflective observations to ensure this stage is performed in a session.

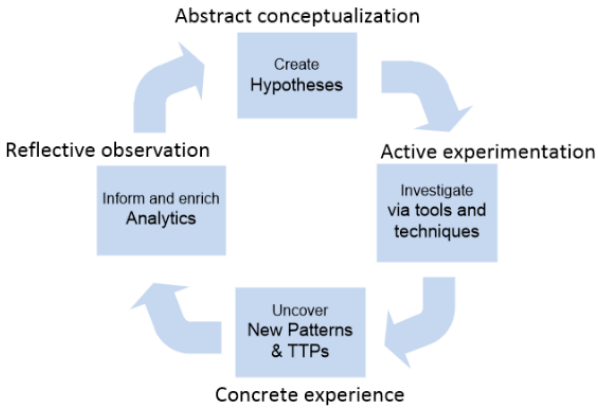

Figure 1. Kolb's experiential learning related to the cyber threat hunting process.

2.1.2 Constructive alignment

Briefly explained, constructive alignment means that pedagogical activities are linked to clear learning goals, and those exposed to it are informed about this. The idea can trace back to Biggs (1996). Constructive alignment is often used in cyber security related courses, labs and exercises. That is, there are typically some efforts to ensure that written objectives are provided, that tasks are related to these objectives, and that assessments or tests measure against the same objectives. A design principle used for Skade is to have each challenge, and part of challenge, be connected to specific learning objectives related to threat hunting. A set of learning objectives for threat hunting was identified and used for this purpose. The learning objectives are described in appendix.

2.1.3 Feedback and assessment

To give students feedback and assessments is also a corner stone in pedagogical best practice. However, not all labs and exercises provide feedback and assessments explicitly. However, it is considered good practice to give feedback and assessments in hands-on cyber security engagements. For instance, the cybersecurity exercise handbook by MSB (2012) recommends after-action reviews of cyber security exercises and concludes that defenders tend to be more motivated if there is some clear system for rewards or scoring. Based on this, frequent feedback was predicted to be important in the Skade web application, primarily because trainees would be working on the challenges alone with no human supervision. Under these circumstances, feedback and assessments of progress may be crucial to get trainees on the right track and keep them from losing motivation.

2.1.4 Motivating setting

Supporting a *motivating setting* is known to be important for learning in general. For instance, Turner & Paris (1995) suggest that motivation in child pedagogy is achieved with six factors ("six Cs"): choice, challenge, control, collaboration, constructing meaning, and consequence. Many of these are central in cyber security exercises, but less prominent in typical hands-on labs. For instance, exercises tend to offer more choices and more opportunities for trainees than hands-on labs, which tend to be designed for specific tools or techniques. In Skade, the focus was placed on choice (e.g., allowing trainees to decide what to focus on), control (e.g., what tools to use) and challenge (e.g., choosing the difficulty level). For instance, a trainee may either try to answer the question from network logs, a central log database, or by logging on to machines and looking at each machine's host logs. All of these alternatives could lead to a successful threat hunt, and giving the trainee control over how they approach the challenge.

2.2 The scenario specification language

The first objective of the project was to create and develop a threat hunting scenario specification language that could be used to instrument challenges. As a part of developing and testing the specification language, three challenge packages were produced, each consisting of two internally related practical challenges:

- **Package A:** Practical threat hunting in a small military network with 30 machines. The first challenge focuses on establishing a network baseline, while the second requires the trainee to conduct threat hunting activities within the network. Both challenges use OpenSearch as the primary analysis tool.
- **Package B:** Practical threat hunting in a small corporate network with 40 machines. Both challenges focus on intelligence-driven threat hunting. The first challenge emphasizes host log analysis and manual work to sift through logs. The second one contrast this with a similar task where the trainee used the SIEM system Wazuh.
- **Package C:** A theoretical package in which trainees is presented with typical threat hunting scenarios and asked to respond to related questions.

To ensure relevance, these three were drawn from already existing scenarios (e.g., exercises and networks) used by FOI, or developed based on instructional material and guides for threat hunting produced by others. They served as a basis for what a scenario specification language used to manage the challenges in a web application.

At the start of each challenge, trainees are provided with a scenario description, relevant network information, and a set of questions to answer. Trainees get access to selected machines within the network through a remote desktop connection. The questions serve as guidance throughout the threat hunting process. For most challenges, there are also hints available that can be used in case the trainee feels the need for it.

In Skade, two files are used to define a scenario: a JSON file and a Python file. The JSON file functions as the scenario specification and contains a set of predefined fields used to provide information to the trainee and to define the interactions with the underlying cyber range and event emulators. The Python file is used to implement scenario specific logic. The connection between the scenario specification, the Python file and the user interface is illustrated in Figure 2.

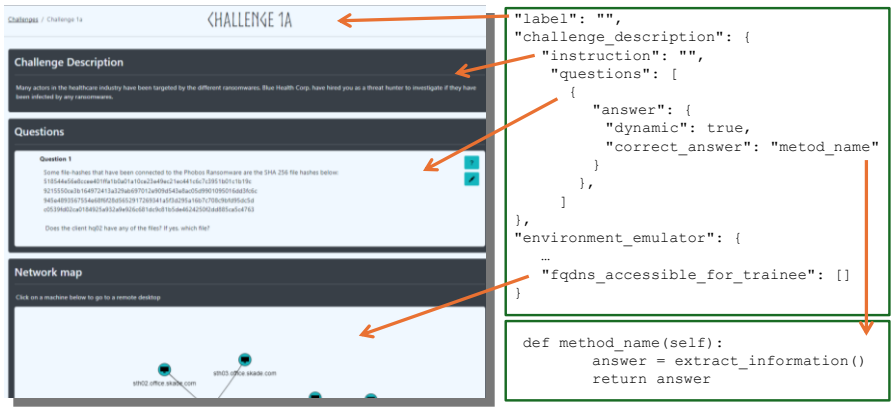

Figure 2: The scenario description and how it relates to the user interface.

The full scenario specification language consists of multiple fields of information. However, to give an understanding of how it is used for defining scenarios we further explain the most important ones, *environment_emulator*, *event_emulation*, and *challenge_description.*

The *environment_emulator* field contains all information related to the cyber range environment. This includes, the name of the environment, the name of the snapshot that the environment should be restored to and the machines accessible to the trainee. Figure 3 shows an example from one of the challenges in package B.

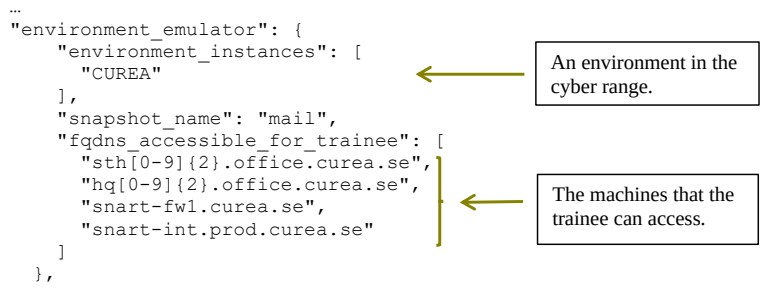

Figure 3: The section detailing the environment emulation segment.

The *event_emulation* field contains information about the events that are executed during the challenge. These events may represent malicious activities, such as attacks performed by a threat actor, or benign network activities, such as actions generated by simulated users. The field describes what emulator to use, if the event sequence is generated during the challenge or not, the name of the event sequence and when to start each event. If the events are generated during the preparation of the challenge, a generation script has to be included for the scenario. Figure 4 illustrates this for one of the challenges in package A.

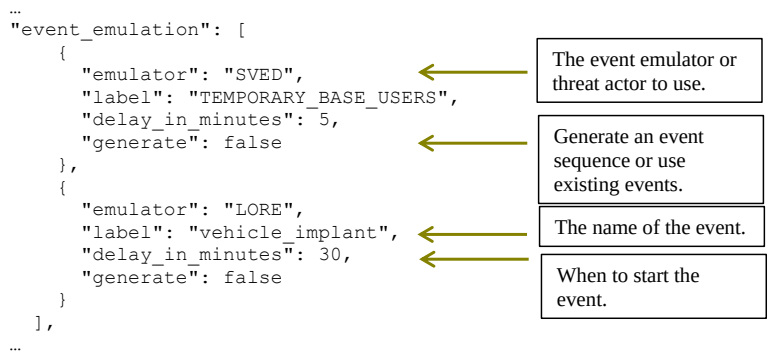

Figure 4: The section detailing the event emulation segment.

The field *challenge_description* describes the challenge and details all questions used in the scenario. This includes the scenario description, the estimated time for the challenge and the questions. Figure 5 shows an example, with strings for instructions and the intro text removed.



Figure 5: The section detailing the challenge description segment.

Each question consists of multiple fields, including a label, question text, dependencies, and learning objectives. A dummy example is presented in Figure 6. The *dependency* field specifies when a question should be presented to the trainee. Each question also has a field, *alternatives*, specifying how the trainee should interact with the question, for example through a textbox or as checkboxes with multiple answers.

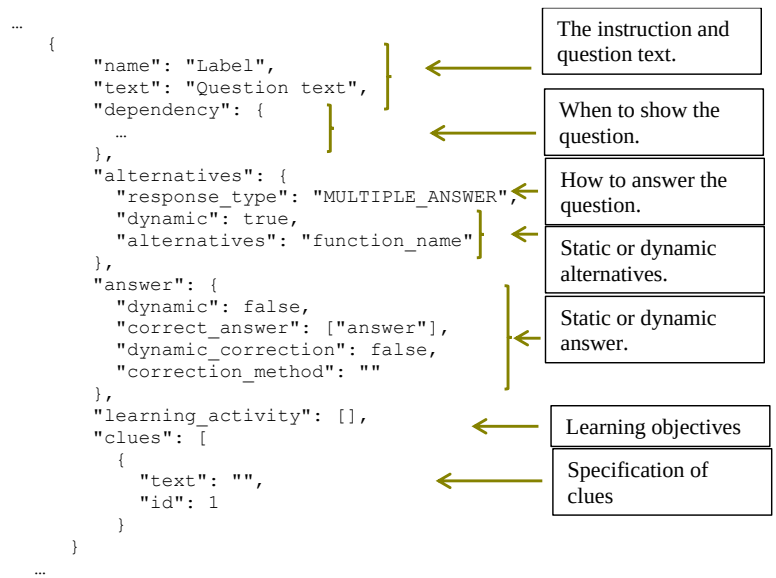

Figure 6. An example detailing a question, with dummy data.

Questions also include an *answer* field that defines how responses are evaluated. Because Skade is designed to handle variations, the *alternatives* and *answer* fields can be either static or dynamic. For example, a static answer may be a pre-defined list of one or more correct answers. This is commonly combined with static alternatives, where some of the alternatives are correct and some are not. In contrast, a dynamic answer is defined by a method name. During challenge execution, the corresponding method is invoked and returns a list of correct answers. An example of a dynamic answer was shown in Figure 2. In that question, the hashes representing response alternatives were generated dynamically with the Python code, and it was ensured that the correct answer was one of the alternatives.

An example of a question that requires dynamic correction is: “Which machines are affected by the malware?” This is because the malware does not spread deterministically, hence compromised machines may differ between challenge executions, and the correct answer is not known before the threat has been simulated. The question evaluation function typically retrieves the logs from the threat emulator platform (e.g., the application programming interface of Lore) and uses this information to determine the correct answer.

2.3 Application design and integrations

The second objective of the project was to develop a prototype called Skade capable of instantiating different threat hunting training scenarios with different properties.

The prototype built in the project utilizes a cyber range and event emulators (threat actors). More specifically, an instance of the cyber range platform Crate is used together with the threat emulators Sved (Holm and Sommestad, 2016) and Lore (Holm, 2023). Figure 7 illustrates an overview of the architecture of Skade.

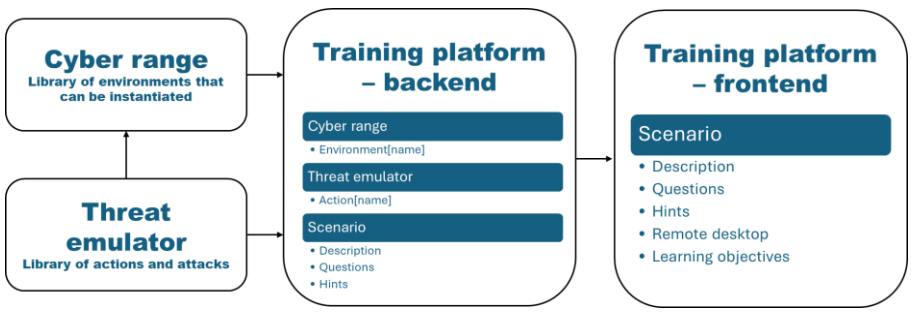

Figure 7: An overview of the architecture of Skade.

Skade relies on the REST APIs provided by the cyber range and tool simulating the threat. This design enables Skade to be integrated with any cyber range or threat emulator implementation that offers similar functionality.

- The cyber range must provide a library of environments Skade can retrieve with a call, the possibility to restore the environment to a start state, and a mechanism for providing browser-based remote desktop access.
- The requirements on the threat emulator are less demanding, as Skade only needs the ability to start and stop events.

While basic functionality is sufficient for integration, more advanced platforms enable the development of more sophisticated and realistic challenges. In particular, to dynamically generate alternatives and responses, live logs are required.

Before a challenge is started, a number of preparatory steps is completed to prepare with simulations. These preparations are initiated when a user books a challenge. As part of the booking, a *start preparation time* is specified. At this time, the cyber range environment is restored to a known state using a snapshot, and with a predefined delay the emulators of benign and malicious events are started. When the booking starts, trainees access the machines through browser-based remote desktop connections. A connection can be established either by selecting a machine in the network map or by clicking the corresponding button in the machine list. Figure 8 illustrates how to establish a connection to a machine.

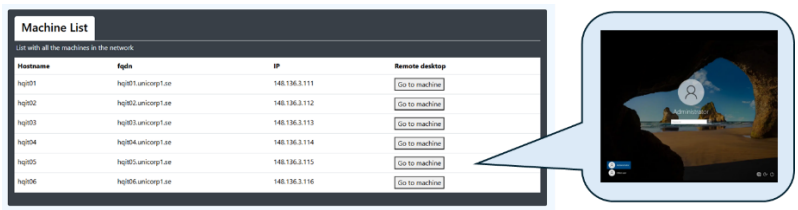

Figure 8: The trainees access the machines through a remote desktop.

3 Field tests

The third objective of the project was to evaluate Skade and the training cases it generates. During 2025, a field test was performed to test the overall usability of the Skade application, and to test four hypotheses:

H1. Satisfaction with difficulty level is associated with higher usability.
H2. Perceived learning is associated with higher usability.
H3. The availability of investigative clues increases satisfaction with difficulty.
H4. The availability of investigative clues increases perceived learning.

The sections below describe the test design, the overall usability (H1 and H2), and the impact of clues (H3 and H4). Further details are available in the paper by Nyholm et al. (2026).

3.1 Test design

The target group for the proof-of-concept challenges was junior threat hunters and system administrators with basic cyber security skills. To recruit this type of users, the Swedish Civil Defence and Resilience Agency (at the time called the Swedish Civil Contingencies Agency) supported the requitement by sending invitations via their stakeholder networks. It had estimated that approximately 40 persons would be needed to test the hypotheses, and 45 participants were recruited.

Each recruited trainee had the option of selecting one of three challenges described in section 2.2. Each challenge was divided into two sessions, where clues were available in one session, as illustrated in Figure 9. It was randomly decided if the trainee would receive the alternative with clues available in the first or second session. Questionnaires were presented before the challenges started, after each session, and after all challenges were completed. The first questionnaire asked the respondent about their background; the questionnaire after each session asked about the previous session; and the final questionnaire asked about the overall experience and about the web application.

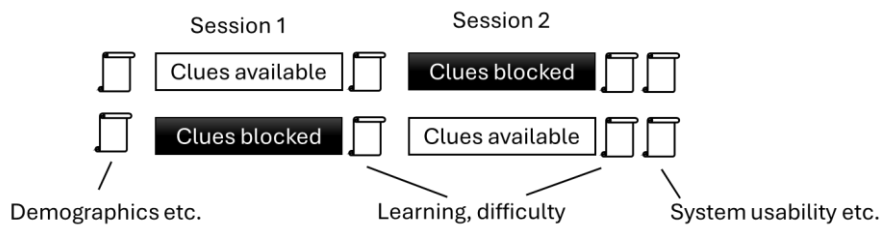

Figure 9. The crossed design of the field test.

The sections below focus on the responses to questions in the questionaries. First, the overall system usability is described. Thereafter, the effects of clues are described.

3.2 Overall system usability

Usability was measured with the widely adopted System Usability Scale in the final questionnaire. This scale consists of ten straightforward questions (e.g., “I think that I would like to use this system frequently”) and responses can be converted to a value 0–100, for which benchmarks exists. On this scale, Skade received an overall mean of 67.2, the range was 17.5–90.0, and the standard deviation was 16.3. The spread in ratings is illustrated in Figure 10. Because the System Usability Scale is widely used, there are many benchmarks for different types of systems. The score 67.2 places the Skade application around typical business-to-business software and public facing websites (Lewis, 2018).

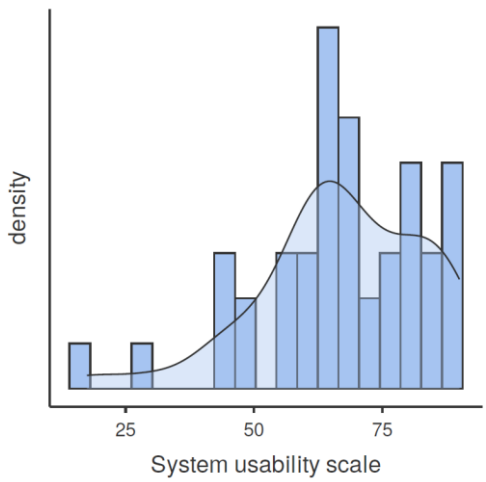

Figure 10. The trainees' ratings of Skade with the System Usability Scale.

Our first hypothesis (H1) was that usability is higher when trainees are satisfied with the difficulty level of the challenges. The second (H2) was that usability is higher when trainees perceive that they learn from challenges. Satisfaction with the difficulty level and the perceived learning were measured twice (after each challenge) using multiple questions with a scale 0-6, where higher is more learning and better difficulty level. The mean of all these responses was correlated with the responses to the overall system usability scale. The results were clear and positive:

- Satisfaction with difficulty level explains 23% of the variation in the System Usability Scale.

- Perceived learning explains 11% of the variation in the System Usability Scale.

Unsurprisingly, these two variables overlap and explain more or less the same variance. When both difficulty satisfaction and perceived learning are entered into a linear regression model, they explain 26% of the variation in responses to the System Usability Scale. Nevertheless, it can be argued that both hypotheses related to system usability are supported by the data. That is:

- Satisfaction with difficulty level is associated with higher usability.
- Perceived learning is associated with higher usability.

The fact that 74% of the variance is not explained by these two suggests that there are many factors other than the difficulty level and the learning experience that influence the perceived usability of the Skade application.

As part of the final questionnaire, the user was asked to give input of issues. These answers could give some insights on other reasons for poorly perceived user experience. Issues reported in the final questionnaire included:

- using the hints (N=8 had serious problems with getting needed help, N=11 had some problems)
- trainees experienced (some, or rarely, serious) problems with the purpose of the challenge questions (N=16)
- access permissions within machines (N=14)
- unresponsive systems (N=11)
- interaction within machines (N=9)
- challenge background information (N=9)
- low performance of the platform (N=8)

The first of these bullets is likely a result of the experimental design, where clues were blocked on purpose. The others indicate poor challenges or technical issues with accessing the cyber range. Thus, issues appear to be related to the underlying emulators and to the challenges as such, not the technical implementation of Skade.

3.3 Effect of clues

In the tests, it was randomized if a participant would receive clues in the first or second session. This was to avoid any systematic bias. Systematic effects are also visible in the data, with the first session being ranked as a better learning experience than the second. The randomization makes sure that the rows at the bottom of Table 1 are not biased by this. As can be seen from the bottom rows in Table 1, there is hardly any difference between sessions with clues and those without.

Table 1. Rating of difficulty satisfaction and perceived learning for different conditions.

Session	Clues	Difficulty satisfaction (mean)	Perceived learning (mean)
First	Yes	4.49	3.89
First	No	4.03	3.66
Second	Yes	3.31	3.47
Second	No	3.35	3.64
Either	Yes	3.86	3.67
Either	No	3.71	3.65

The ratings are similar whether clues are available or not. Hence, the two hypotheses related to clues are not supported in the data.

In other words, it is **not** the case that:

- The availability of investigative clues increases satisfaction with difficulty.
- The availability of investigative clues increases perceived learning.

This was somewhat surprising, especially since many trainees also highlighted that they would have wanted more help with the type of technical questions made available in the clues (e.g., queries used to search for logs). We discuss this further below.

4 Discussion and future work

The project *Skade: a scenario generator for threat hunting* had three main objectives:

- To create a threat hunting scenario specification language.
- Develop a prototype that instantiates the scenarios in a cyber range.
- To evaluate the prototype and the challenges it generates.

The sections above describe how the project meets these objectives. The field test with external trainees that received a decent usability rating (cf. Figure 10) is central. However, there are also some limitations:

- The scenario specification language is limited to the types of challenges in the field test. Extensions may be required in the future, e.g., to handle challenges with participants acting in a team.
- The prototype did not implement trainee progress management, and is only integrated with FOI's own platform Crate and its support tools. However, we believe it can be easily integrated with other platforms too. For instance, a scenario designer could instead integrate the threat emulator Caldera[8] by writing Python code to interact with its application programming interface.

The evaluation of the prototype was successful, but also showed some surprises. The biggest surprise was the non-effect of clues. The prototype was designed to balance difficulty levels using clues, and the field test of the prototype was designed to measure the effect that clues had on trainees' difficulty satisfaction and perceived learning. Our retrospective analysis finds no obvious reason why clues did not help. For instance, in the questionnaires several participants asked for technical support similar to those provided in the clues. However, we found that 70 percent of difficulty satisfaction has to do with the persons doing the challenge, not how the challenge is constructed. Thus, only 30 percent of the satisfaction with the difficulty level was influenced by clues, questions, response alternatives, background text, the actual threat, and the time available. We speculate that this is because:

- Some individuals navigate technical problems well by searching the internet for technical assistance, regardless if clues are made available or not.
- Some individuals have a hard time understanding or using the information provided in the clues because they lack some basic skill required in the threat hunt, and the clues do not help.

[8] https://github.com/apache/caldera

The second biggest surprise of the project was the stability of the technical integration with the cyber range. Even though a dozen independent components are involved in each scenario execution, problems seldom emerged. When they did, they were often due to poor configuration control, e.g., because a threat emulator had been automatically updated and there had been changes on how to interact with it. Few of the 45 trainees participating in the field test experienced any serious issues originating from the platform Crate or its support systems.

We believe further development of Skade, or other facades to cyber ranges, can produce dynamic learning systems where experts and novices can be challenged with the right difficulty level in a progressive manner, and learn important tradecraft, before they need to deal with real incidents. There are several things that are needed to make Skade, or similar systems, suitable for producing strong learning outcomes at scale. We anticipate that the following are needed:

- More robustness in the technical platform, both concerning the web application and the version control of the emulators used by the application.
- Progress management, or other ways to characterize and predict trainees' proficiency levels.
- A library of different challenges, targeting various specialties and roles, and with some system of predicting or manipulating difficulty level.
- Clues and support delivered in other ways, such as chatbots that can answer different types of questions.

Efforts along these lines are already financed for 2026–2027.

5 References

Beuran, R., Tang, D., Tan, Z., Hasegawa, S., Tan, Y., Shinoda, Y., 2019. Supporting cybersecurity education and training via LMS integration: CyLMS. Educ. Inf. Technol. (Dordr). 24, 3619–3643.

Biggs, J., 1996. Enhancing teaching through constructive alignment. High. Educ. (Dordr). 32, 347–364.

Cyberbit, 2020. Cyberbit Range: Assessing Cyber Skills to Improve SOC Team Performance.

Holm, H., 2023. Lore a Red Team Emulation Tool. IEEE Trans. Dependable Secure Comput. 20, 1596–1608.

Holm, H., Sommestad, T., 2016. Sved: Scanning, vulnerabilities, exploits and detection. In: MILCOM 2016-2016 IEEE Military Communications Conference. pp. 976–981.

Kolb, D.A., 1984. Experiential Learning experience as a source of learning and development. Prentice Hall, New Jersey, US.

Lewis, J.R., 2018. The System Usability Scale: Past, Present, and Future. Int. J. Hum. Comput. Interact. 34, 577–590.

Miazi, M.N.S., Pritom, M.M.A., Shehab, M., Chu, B., Wei, J., 2017. The design of cyber threat hunting games: A case study. In: 2017 26th International Conference on Computer Communications and Networks, ICCCN 2017.

MSB, 2012. Handbok – Tekniska informations- och cybersäkerhetsövningar. Stockholm, Sweden.

Nyholm, L., Sommestad, T., Karlzén, H., Kvist, H., 2026. Managing Learning and Difficulty Satisfaction in Cybersecurity Training Challenges. In: European Interdisciplinary Cybersecurity Conference 2026. Springer Nature, Darmstadt, Germany.

Petersen, R., Santos, D., Smith, M.C., Wetzel, K.A., Witte, G., 2020. Workforce Framework for Cybersecurity (NICE Framework). Gaithersburg, MD.

Pfaller, T., Skopik, F., Smith, P., Leitner, M., 2024. Towards Customized Cyber Exercises using a Process-based Lifecycle Model. In: ACM International Conference Proceeding Series. Association for Computing Machinery, pp. 37–45.

Seda, P., Vykopal, J., Svabensky, V., Celeda, P., 2021. Reinforcing Cybersecurity Hands-on Training with Adaptive Learning. In: Proceedings - Frontiers in Education Conference, FIE. Institute of Electrical and Electronics Engineers Inc.

Sommestad, T., Karlzén, H., Kvist, H., Gustafsson, H., 2023. Skade – A Challenge Management System for Cyber Threat Hunting. In: 9th Workshop On The Security Of Industrial Control Systems & Of Cyber-Physical Systems. Haag, Netherlands.

sqrrl, 2018. A Framework for Cyber Threat Hunting.

Turner, J., Paris, S.G., 1995. How literacy tasks influence children's motivation for literacy. Read. Teach. 48, 662–673.

Appendix: Learning objectives

Stage(s)	Learning objective: The student shall be able to…
Hypothesis development	In a given network, suggest a logical attack chain and timeline, e.g. think like a generic threat agent or like a particular threat agent.
	In a given network, consider the implications of threat intelligence, e.g. what hosts it would pertain to.
	In a given network, consider the implications of alerts and security warnings and what they may indicate, e.g. if they imply attackers may have access.
	Determine if a threat hunting effort is warranted or worthwhile, e.g. if the risk is big enough, how often, how much time to spend, when to stop, and rules of engagement.
	Formulate testable hypotheses applicable to threat hunting, e.g. the logs that would be present due to an attack.
Investigate via tools and techniques	Find and query events in log databases, e.g. SIEM systems.
	Find and query events in networks, e.g. PCAP and netflow from gateways.
	Find and query events on hosts, e.g. syslog and windows events.
	Find and query data from of various kinds using scripts or tools, e.g. different formats from with different data schemas.
	Perform triage: identify, prioritize, and resolve cybersecurity incidents.
Uncover new patterns and TTPs	Recognize events or artifacts that match a specific IoC, e.g. an IP or file hash.
	Recognize events and artifacts that relate to threat agents' activities, e.g. events related to a tool or new vulnerability.
Inform and enrich analytics	Assist construct signatures for implementation in tools such as SIEM systems.
	Work together and communicate with other threat hunters and SOC personnel etc.
	Assess risks and suggest security improvements.
	Account for conducted work in communication with stakeholders.

FOI
Totalförsvarets forskningsinstitut
164 90 Stockholm

Tel: 08-55 50 30 00
Fax: 08-55 50 31 00

www.foi.se