Education and Information Technologies (2022) 27:9231–9262
https://doi.org/10.1007/s10639-022-10954-4

# Student assessment in cybersecurity training automated by pattern mining and clustering

**Valdemar Švábenský[1,2] · Jan Vykopal[1] · Pavel Čeleda[1] · Kristián Tkáčik[2] · Daniel Popovič[2]**

Received: 22 October 2021 / Accepted: 9 February 2022 / Published online: 30 March 2022


## Abstract

Hands-on cybersecurity training allows students and professionals to practice various tools and improve their technical skills. The training occurs in an interactive learning environment that enables completing sophisticated tasks in full-fledged operating systems, networks, and applications. During the training, the learning environment allows collecting data about trainees' interactions with the environment, such as their usage of command-line tools. These data contain patterns indicative of trainees' learning processes, and revealing them allows to assess the trainees and provide feedback to help them learn. However, automated analysis of these data is challenging. The training tasks feature complex problem-solving, and many different solution approaches are possible. Moreover, the trainees generate vast amounts of interaction data. This paper explores a dataset from 18 cybersecurity training sessions using data mining and machine learning techniques. We employed pattern mining and clustering to analyze 8834 commands collected from 113 trainees, revealing their typical behavior, mistakes, solution strategies, and difficult training stages. Pattern mining proved suitable in capturing timing information and tool usage frequency. Clustering underlined that many trainees often face the same issues, which can be addressed by targeted scaffolding. Our results show that data mining methods are suitable for analyzing cybersecurity training data. Educational researchers and practitioners can apply these methods in their contexts to assess trainees, support them, and improve the training design. Artifacts associated with this research are publicly available.

**Keywords** Cybersecurity education · Security training · Data science · Educational data mining · Learning analytics

✉ Valdemar Švábenský
svabensky@ics.muni.cz

Extended author information available on the last page of the article

## 1 Introduction

Cybersecurity professionals are needed across the globe to counter the ubiquitous cyber threats. To meet the increasing demand for cybersecurity experts ((ISC)2, 2021), effective training is essential. Such training must include hands-on components and provide practical experience in authentic settings. This includes using a variety of tools for cybersecurity operations, such as host configuration, hardening, and penetration testing.

Cybersecurity experts use tools with graphical user interfaces as well as command-line tools. Within the scope of our research, we focus on the latter, since command-line tools represent an important component of cybersecurity practice. Cyber attackers use them to perform sophisticated attacks, which cyber defenders need to understand to mitigate advanced threats. In addition, various command-line tools are used to configure computer systems securely.

### 1.1 Research problem statement

Our research focuses on *supporting automated assessment in the context of hands-on cybersecurity training*. Here, we explain the motivation for our research, illustrate the problem with a simple example, justify why the problem is hard to address, and summarize the gaps in the current literature.

**Why is student assessment necessary?** Educational assessment is a crucial aspect of training (Lancaster et al., 2019). It enables teachers (*instructors*) to better understand the actions of their students (*trainees*). Specifically, in-depth assessment shows what each student did well, what could be improved, and whether the student progressed through the training as expected.

Based on insights from the assessment, teachers can adapt their class, provide students with feedback to support their learning, or evaluate their level of knowledge. The assessment also shows potential issues in the training design, enabling to fix them and further improve the effectiveness of the training.

**How can students be assessed?** Like most applied computing skills, cybersecurity is usually practiced hands-on in computer-supported interactive learning environments. These are physical or virtual platforms that provide computer hosts with full-fledged operating systems, networks, and applications for training.

Advanced interactive learning environments allow collecting data about the students' actions, such as their usage of command-line tools. These student interaction data authentically capture learning processes. Therefore, they can be transformed into educational insights and exploited for assessment.

As an example, consider the two command histories from cybersecurity training shown in Figs. 1 and 2. They belong to two students who attempted to crack a password to a ZIP archive using the `fcrackzip` utility in Linux. Each command is prefixed by the timestamp of its execution. Based on the analysis of these student

```
12:27:41 fcrackzip -b -u file.zip
12:28:25 fcrackzip -b -D -u file.zip
(repeated 5 times)
12:29:17 fcrackzip -b -u /file.zip
12:29:24 fcrackzip -b -u
12:29:34 fcrackzip -b -u /file.zip
12:30:13 fcrackzip -D -u -v file.zip
12:30:51 fcrackzip -D -u -v file.zip
12:30:55 fcrackzip -b -u -v file.zip
12:31:18 fcrackzip -c -u -v file.zip
(9 more executions with different option combinations)
12:33:02 fcrackzip -b -u -v file.zip
12:37:22 fcrackzip -v -u -D fasttrack.txt file.zip
```

**Fig. 1** The first student ran the cracking tool 24 times within an approximately 5-minute time frame, with various combinations of arguments, often repeating the previous (incorrect) combinations. After that, the student stopped for 4 minutes, probably to find help, and executed a correct command

```
20:48:49 fcrackzip.exe-help
20:48:55 fcrackzip.exe -help
20:49:02 fcrackzip.exe --help
20:49:04 fcrackzip
20:49:08 --help
20:49:13 fcrackzip --help
20:50:44 sudo apt-get install fcrackzip
20:52:51 fcrackzip -u -D -p fasttrack.txt file.zip
```

**Fig. 2** The second student assumed that the tool had the `.exe` suffix of Windows OS executables, which does not apply to Linux OS. The student was apparently unfamiliar with Linux or the cracking tool but then instantly executed a correct command without any previous incorrect tries. We can assume that they received outside help

data, the instructor can see that each student needs help with a specific and different aspect of the training.

**Why is assessment difficult?** In-depth assessment of cybersecurity training is difficult for four main reasons.

1. *The training is complex.* The tasks require high-order problem solving and may have many different correct solutions. Therefore, the assessment is much more complex than assessing simple tasks such as memorizing facts.
2. *Each student is unique.* Every student has different previous knowledge, experience, motivation, and approach to learning. As a result, students adopt different strategies to solve the tasks. This is natural, but it further complicates the conditions for automatically assessing hands-on tasks.
3. *Students generate a lot of data.* During the training, even a class that is relatively small (10–20 students) and time-constrained (1–2 hours) can generate hundreds of data records. As a result, manually processing these data becomes quickly infeasible.
4. *The assessment process is not straightforward.* It is unclear how to transform the raw data from training into educational insights useful for assessment. As the

examples in Figs. 1 and 2 demonstrated, even a relatively constrained assignment can generate various data for assessment.

**The need for research** Traditionally, educational researchers and practitioners assessed student data manually. However, due to the difficulties described above, a manual transformation of hands-on training data into educational insights is not viable (Fournier-Viger, 2017; Romero & Ventura, 2020). It is highly time-consuming, ineffective, and error-prone.

Automated assessment is more scalable and accurate. Therefore, it can be fruitful to leverage automated techniques, such as machine learning and data mining, for analyzing data from hands-on training (Palmer, 2019). These techniques should transform the data from their raw form to an understandable representation, such as an overview of highlights or a visualization.

However, the review of current literature (see Section 3 for details) identified several gaps in state of the art in this area:

- As Weiss et al. (2016) argued, current automated assessment is often superficial, judging only the (in)correctness of the solution. Only a few papers, such as by Mirkovic et al. (2020), have explored an in-depth assessment of student learning.
- To the best of our knowledge, no published research attempted to compare and evaluate the applicability of two different data mining methods on cybersecurity training data. Student assessment in cybersecurity has been explored from other perspectives, such as using numerical scoring metrics (see Maennel et al. (2017) for an example).
- Data mining algorithms have been used for assessment in other domains, such as programming (Gao et al., 2021), but it is unclear how to generalize these previous results to the cybersecurity context.

### 1.2 Goals of this research paper

We seek to support automated assessment of students in hands-on training. In order to address the gaps in the literature, the assessment must satisfy the following criteria:

- enable an in-depth understanding of students' actions,
- use methods that have not been researched in this context previously, and
- be evaluated on an authentic dataset from realistic training sessions.

The domain of data mining offers many methods for the automated extraction of insights from raw data (Fournier-Viger, 2017). Two methods that satisfy the criteria above and will be explored in this paper are *pattern mining* and *clustering*. Pattern mining techniques, such as association rule mining and sequential pattern mining, can reveal interesting relationships in datasets (Fournier-Viger, 2013b). Clustering, on the other hand, forms groups of data based on their similar characteristics

(Romero et al., 2010). Evaluating these two techniques represents an original contribution to cybersecurity education and beyond.

**Research questions** Our research is framed by two research questions related to student assessment in cybersecurity: *What insights can we gather from command histories using pattern mining* (RQ1) *and clustering* (RQ2)*?* By *insights*, we mean the following educational findings to support assessment:

- trainees' approaches and strategies to solving the training tasks,
- common mistakes, misconceptions, and tools problematic for trainees,
- distinct types of trainees based on their actions and behavior, and
- issues in the training design and execution.

**Expected contributions of this research** Answering the research questions will be valuable for various stakeholders.

- *Cybersecurity instructors* can use the researched methods in their classes to gain new insights for assessing their students. Specific assessment use cases are detailed in Sections 5.3 and 5.5.
- *Researchers* can build upon this work by evaluating other data mining methods on similar datasets. This will contribute to the body of knowledge on assessment in cybersecurity training.
- *Developers of cybersecurity training platforms* can integrate the researched methods of data collection and analysis into the interactive learning environments. This will support the goals of instructors and researchers.

Educational stakeholders from outside the cybersecurity domain can benefit from this research as well. Students of related computing disciplines, such as networking and operating systems administration, can generate similar data for assessment in hands-on classes. For students of other disciplines, the researched methods can be extended to process different data, such as clickstreams.

### 1.3 How to read this paper

Above, we defined three target groups who may be interested in this paper. Although we aim to address readers from a broad audience, we acknowledge that some sections of the paper are not relevant for everyone. Section 2 provides a brief background and therefore aims at *researchers* who seek to understand the theory of the used methods. Other readers who are satisfied with a more high-level understanding may skip it. Section 3 reviews related studies, which is relevant for *researchers* and *instructors* interested in how the previous research results were applied to support teaching practice. Section 4 details the used methods for the data collection and analysis. It is aimed mainly at *researchers* and *developers*, since it also includes technical details about the training platforms and data collection. Section 5 presents

the findings and answers the research questions. Finally, Section 6 concludes, summarizes our contributions, and proposes future work. These two sections are suitable for all readers.

## 2 Background and terminology of data mining

This section defines the key terms to familiarize the readers with basic data mining concepts. *Data mining* is a field of computing that deals with extracting knowledge from data. Its purpose is to enable understanding of the data, gather new insights from them, and support decision-making based on this understanding (Fournier-Viger et al., 2017; Han et al., 2011). Out of the many data mining methods, we will focus on two of them: pattern mining (Section 2.2) and clustering (Section 2.3).

### 2.1 Educational data mining and learning analytics

*Educational data mining* (EDM) (Romero et al., 2010) and *Learning analytics* (LA) (Lang et al., 2017) are two inter-related research areas that aim to understand and improve teaching and learning. The research in these areas focuses, for example, on student behavior, learning processes, assessment, and interactive learning environments. To achieve their aims, EDM/LA researchers collect and analyze data from educational settings.

### 2.2 Pattern mining

*Pattern mining* automatically extracts previously hidden patterns in data. Its objective is to discover patterns that are easily interpretable by humans. We concentrate on two well-established pattern mining techniques: association rule mining (ARM) and sequential pattern mining (SPM) (Fournier-Viger, 2013b; Fournier-Viger et al., 2017).

**Association rule mining** *Association rules* are patterns with the form of an *if-then* statement. A rule $X \rightarrow Y$ says that if an *item* $X$ occurs in a *transaction* (a set of items), then so does $Y$ (Fournier-Viger et al., 2017; Han et al., 2011; Romero et al., 2010). In our case, an item may be a command submitted by a student, and a transaction may be a whole set of commands of that student. An association rule mined from a set of students' transactions may indicate that if a student used a command $X$, then they used a command $Y$.

For each association rule $X \rightarrow Y$, we are typically interested in two metrics: its *support* (relative occurrence among all the examined transactions) and *confidence* (relative occurrence among the transactions that contain $X$).

Algorithms for mining association rules consider only rules that satisfy the user-defined thresholds for the minimal support and confidence, `MinSup` and `MinConf`. Since this process can extract a vast amount of rules, additional measures such as *lift*

are applied to filter out irrelevant rules (Fournier-Viger et al., 2017; Han et al., 2011; Romero et al., 2010).

**Sequential pattern mining** *Sequential pattern* is a frequently occurring subsequence in a given set of sequences (Fournier-Viger et al., 2017; Romero et al., 2010). For example, it can be a progression of certain commands that many students used. Contrary to ARM, SPM can analyze data in which the ordering of items is relevant.

Again, sequential patterns are mined based on a `MinSup` threshold. To find a manageable amount of patterns, it is recommended to use algorithms that mine *closed sequential patterns* (Fournier-Viger et al., 2017; Fournier-Viger et al., 2014; Fumarola et al., 2016).

### 2.3 Clustering

*Clustering* is the process of assigning data points into groups called *clusters* based on their similarity. Data in one group are similar to each other and dissimilar to data from other groups (Madhulatha, 2012). For example, in our context, we can group students based on the similarities in their command-line usage. Clustering is an unsupervised machine learning technique, so it does not use previously labeled data to assess new data. Instead, it organizes unlabeled data into "bundles".

We focus on *density-based* clustering, which defines a cluster as an area with a high density of data points; low-density areas separate individual clusters. Unlike *partitional* clustering methods, such as the popular *k*-means clustering (Lloyd, 1982), density-based approaches are better at recognizing arbitrarily shaped clusters and filtering noise or outliers. However, not all data points may end up in a cluster (Beyer et al., 1999; Aggarwal et al., 2001).

## 3 Related work

This section reviews the publications related to the analysis of educational data. It also explains how our research differs from state of the art.

### 3.1 Pattern mining in educational data

Association rule mining (ARM) or sequential pattern mining (SPM) has been employed to investigate various aspects of education. These include learner difficulties, correlations between learning behaviors and performance, and teaching strategies that lead to better learning (Romero and Ventura, 2020; Bienkowski et al., 2012).

García et al. (2010) applied ARM on data capturing students' usage of a learning management system, discovering relationships between students' activities and final grades. Instructors can use this information to adjust the course or identify struggling students early. Kobayashi (2014) also used ARM to uncover the errors

that frequently co-occurred at various proficiency levels when learning spoken English. The pattern mining revealed types of mistakes that distinguish lower-level and upper-level students.

Malekian et al. (2020) applied SPM on data representing students' actions and task submissions in an online learning environment. The researchers wanted to discover the behavior patterns that lead to successful or unsuccessful assessment outcomes. Therefore, they split the sequences of actions into two categories depending on the outcome of the sequence's final submission. The failed sequences contained mainly repeated assessment submissions and discussion forum views. In contrast, the passed sequences included multiple reviews of lecture materials. This information can be used to modify the learning environment to discourage unproductive behavior.

Gao et al. (2021) mined sequential patterns from programming logs to identify struggling students. Timely recognizing these students is essential for promoting their learning. To establish ground truth, the researchers again split the logs of high- and low-performing students. Then, they mined patterns that either dominated in one group to discover its specifics, or occurred in both groups to reveal similarities. After that, they used the patterns as features in a classifier algorithm to predict student performance.

### 3.2 Clustering of educational data

Vellido et al. (2010) motivate the usage of clustering in educational contexts. In addition, they also provide a brief overview of literature where clustering was applied to solve educational problems. Next, Romero and Ventura (2010) and Dutt et al. (2017) performed literature reviews of EDM papers. Clustering has been used to provide feedback to instructors, detect undesirable or unusual student behavior, analyze and model student behavior, and group students by various characteristics, such as their learning approaches.

Yin et al. (2015) used the OPTICS algorithm to cluster students' programming assignments, aiming to support autograding based on the type of solution. Student source code was represented as an abstract syntax tree, with the normalized tree edit distance as the similarity measure for clustering. The researchers discovered clusters corresponding to distinct types of solutions (canonical, correct but longer code, complex solution, and so on).

McBroom et al. (2016) mined submission logs from an autograding system for program code. They clustered weekly submissions to find approaches to each assignment while also analyzing the long-term behavior to learn how students develop. The researchers detected common behavioral patterns as early as in week three of the semester, and students' behavior largely remained the same. Teachers can use the gained insight to intervene when a student belongs to the cluster with a higher risk of failure.

The goal of Piech et al. (2012) was to study how students learn to program. To do so, the researchers captured and clustered temporal traces of student interactions with a compiler. They applied a hidden Markov model to the temporal traces and

visualized it as a state machine for the cluster. The model then predicted student performance.

Emerson et al. (2020) explored novices' misconceptions in block-based programming. The researchers used logs of unsuccessful student attempts at programming assignments. The students' programs were represented by three families of features: basic block features, counts of specific block sequences, and the number of interactions with the system. The results revealed three clusters of students: exploratory, disorganized, and near-miss.

In their follow-up work, Wiggins et al. (2021) analyzed novices' hint requests in block-based programming. When a student asked for a hint, the time elapsed from the assignment's start and the percentage of code completion were recorded. Clustering of this data revealed five different groups of students based on their hint-taking strategies. For example, those that asked for a hint early and had low code completeness probably needed a "push" to start. Instructors can use this information to target the students' needs specific to the given group.

### 3.3 Using data for student assessment in cybersecurity

Maennel (2020) performed a thorough literature review of data sources that can serve as evidence of learning in cybersecurity exercises. These data sources include timing information, command-line data, counts of events, and input logs. Our paper investigates the applicability of *command-line data* in educational assessment. Such data are collected in multiple state-of-the-art learning environments for cybersecurity training (Weiss et al., 2017; Andreolini et al., 2019; Labuschagne and Grobler, 2017; Tian et al., 2018).

Weiss et al. demonstrated that command-line data from cybersecurity training are valuable for student assessment. They incorporated information about the students' exact steps, rather than just a numerical score indicating success or failure. They analyzed the students' work processes and the utilized command-line tools. Based on the command histories, they generated progress models of student approaches (Weiss et al., 2016; Weiss et al., 2017; Švábenský et al., 2022) and predicted their success (Vinlove et al., 2020).

Mirkovic et al. (2020) collected and analyzed command-line input and output from participants in hands-on cybersecurity exercises. The analysis system automatically compared the collected data with pre-defined exercise milestones and produced statistics about the participants' progress. It helped identify difficult sections of the exercises and students needing assistance, providing useful information to instructors.

Abbott et al. (2015) parsed a dataset of logs from cybersecurity training into meaningful blocks of activity and statistically analyzed them. McClain et al. (2015) further explored this dataset combined with questionnaires measuring the participants' experience in cybersecurity. They discovered that more experienced participants used specialized and general-purpose tools, while the less experienced participants focused only on specialized cybersecurity tools.

Finally, several works investigated the assessment of teams in sophisticated cyber defense exercises. Granåsen and Andersson (2016) collected network and system logs to study the performance of teams. Similar data sources were used by Henshel et al. (2016) to assess and predict team performance. Maennel et al. (2017) proposed a systematic approach: a methodology to employ exercise data for team assessment. In contrast, we focus on individual assessment during exercises in the scope of classroom teaching.

### 3.4 Summary of the related work

Pattern mining and clustering were applied in educational contexts with interesting results. They can reveal students' misconceptions, approaches to solving the tasks, and behavioral patterns. These insights can improve educational assessment and feedback and target instruction to support students' needs.

The novelty of our paper is exploring these methods in the context of cybersecurity training. Previously, command-line data from cybersecurity training were analyzed using other methods, such as statistics, regular expression matching, and classifiers. We seek to discover insights gathered from cybersecurity training data using pattern mining and clustering, as well as demonstrate their usefulness for assessment. Moreover, we aim to uncover in-depth insights, not only assess the correctness of the student solution.

## 4 Research methods

This section explains the methods chosen to answer the research questions posed in Section 1.2. A visual overview of these methods is provided in Fig. 3. In previous projects (Tkáčik, 2020; Popovič, 2021), we prototyped the methods on smaller datasets, yielding initial results that we updated for this paper.

### 4.1 Cybersecurity training

Our research analyzes data from cybersecurity training. Specifically, we focus on offensive security skills training in a sandboxed network emulated within an interactive learning environment. The following text introduces essential aspects of the training to provide context for the research.

**Interactive learning environment** The virtual machines for the training were hosted in KYPO Cyber Range Platform (Masaryk University, 2021; Vykopal et al., 2021), which is a cloud-based infrastructure for emulating complex networks. For some training sessions, we alternatively used Cyber Sandbox Creator (Masaryk University, 2022a; Vykopal et al., 2021): a tool for creating lightweight virtual labs hosted locally on the trainees' computers. This choice of the underlying infrastructure did not affect the training content, and the data collection was also equivalent.

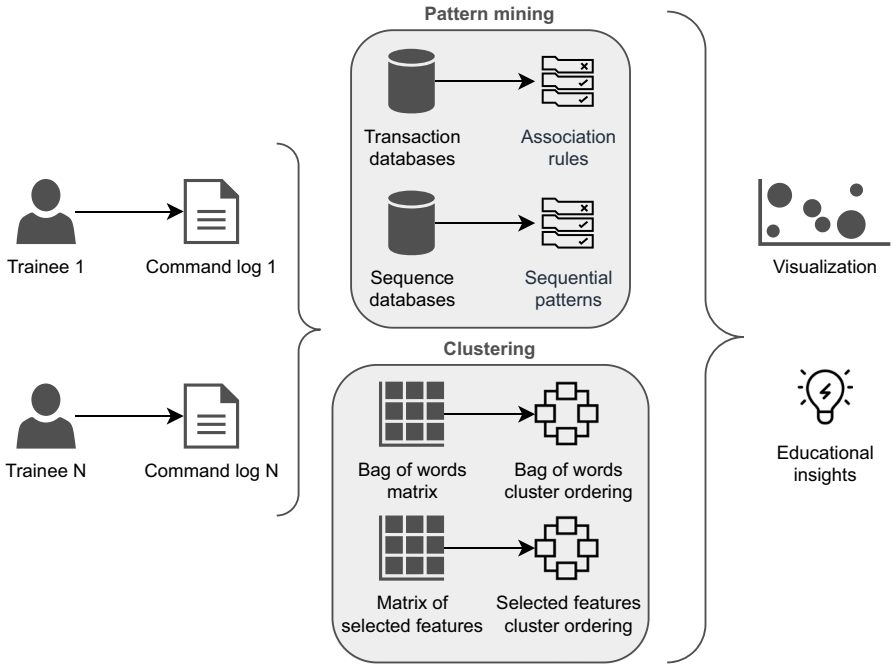


**Fig. 3** The command logs collected from students act as input for pattern mining and clustering. The results are visualized and interpreted in Section 5

Both platforms are open-source (Vykopal et al., 2021), and cybersecurity instructors can freely deploy them for their purposes.

**Training format** The trainees worked with the interactive learning environment either remotely via a web browser or locally on their computers. Each trainee accessed their own isolated sandbox containing a virtual machine with Kali Linux (Offensive Security, 2022a): an operating system distribution tailored for penetration testing that provided the necessary tools. The trainees completed a sequence of assignments presented via a web interface. Almost all the assignments were solved using command-line tools, which are described below.

The participants were allowed to use any sources on the Internet. Moreover, the interactive learning environment offered optional hints, which the trainees could reveal to get help with the current task. The usage of hints and outside help was allowed since the trainees were not evaluated summatively (that is, the training was not a graded exam). Instead, we focused on formative assessment and helping the students explore new cybersecurity skills.

**Training content** Each trainee participated in exactly one of two types of training. Both trainings involved attacking an intentionally vulnerable virtual host using well-known security tools, but the trainings slightly differed in their content. In Training

A (72 participants), the following tools were crucial: `nmap` for network scanning, Metasploit for exploitation, `john` for password cracking, and `ssh` for remote connection. Training B (41 participants) used `nmap` and `ssh` as well, but not Metasploit or `john`. Instead, it featured `fcrackzip` for cracking passwords to ZIP files (see Figs. 1 and 2). None of the trainees was previously familiar with any of these two trainings.

Again, the training content is publicly available (Masaryk University, 2022b). Training A corresponds to the cybersecurity game *Secret laboratory* and its derivatives, while Training B corresponds to the game *Junior hacker training*. Cybersecurity instructors can freely deploy these games in their classes and recreate the conditions for our research.

**Training participants** From August 2019 to February 2021, we hosted 18 cybersecurity training sessions for a total of 113 trainees. Each training session usually took two hours to complete, and most of them were held remotely due to COVID-19 restrictions. The participants included:

- undergraduate and graduate students of computer science from various European universities,
- high school students attending the national cybersecurity competition, and
- cybersecurity professionals.

They all attended voluntarily because of their interest in cybersecurity and were not incentivized. Although the participants do not form a random sample, we argue that it is practically infeasible to recruit a randomized population for this type of research. Therefore, we instead worked with the representatives of the target group for this cybersecurity training.

**Ethical and privacy-preserving measures for research** Since we carried out research with human participants, we ensured that the trainees would not be harmed in any way. We minimized the extent of data collection to gather only the data necessary for the research. We also received a waiver from our institutional ethical board since we do not collect any personally identifiable information.

The participants provided informed consent to the collection and usage of their data for research purposes. The collected data were thoroughly anonymized not to reveal the trainee's identity. As a result, it is impossible to track the trainee throughout future training sessions.

### 4.2 Data collection

While the trainees solve the assignments, our infrastructure (Švábenský et al., 2021) automatically collects their submitted commands and the associated metadata. We gathered data from command-line tools in the Linux Bash terminal and Metasploit shell, which is software for penetration testing (Offensive Security,

2022b). These data, which are published (along with other training data) in an open-source article (Švábenský et al., 2021), serve as the input for pattern mining and clustering. We did not collect data from tools with a graphical user interface.

**Data format** The command history of each trainee is captured in a single JSON file. The file consists of dozens of log records (78 per trainee on average), such that each record represents a single command executed by the trainee. Figure 4 shows an example of such a log record.

Each log record has a fixed number of attributes. For our purposes, the most significant are:

- `timestamp`, representing the time of the command's execution in the ISO 8601 format,
- `cmd`, which represents the full command (the tool and its arguments) submitted by the trainee, and
- `cmd_type`, the application used to execute the command: either "bash-command" for the tools executed within Linux Bash terminal, or "msf-command" for Metasploit shell.

**Data properties** We collected 8834 commands, which constitute the dataset for this research, over the period of 1.5 years. Although this sample is not massive in volume, it captures the trainees' interactions deeply and over prolonged periods. Therefore, it fulfills the prerequisites of the chosen data mining methods.

Hands-on cybersecurity training is usually held in a group of lower tens of participants. Therefore, we consider the 8834 commands to be sufficient for evaluating the two data mining methods. On average, this dataset corresponds to 78 commands per trainee within the 1–2-hour time frame, which is appropriate for the chosen training format.

For this research paper, we focus on data processing after the training ends. Nevertheless, the used methods are applicable during the training for real-time assessment as well.

**Fig. 4** A single log record from a command history of one trainee

```
{
    "timestamp"   : "2020-07-03T08:09:25+01:00",
    "username"    : "root",
    "hostname"    : "attacker",
    "ip"          : "10.1.135.83",
    "sandbox_id"  : "1",
    "wd"          : "/home",
    "cmd"         : "nmap --help",
    "cmd_type"    : "bash-command"
}
```

### 4.3 Pattern mining

To enable mining patterns from the command-line data, our analysis scripts written in Python automatically transformed the input data into the *transaction* and *sequence* databases described below. These databases are an internal representation of the input data, and they serve as the input for ARM and SPM algorithms, respectively. A key advantage of pattern mining is that the data preparation is the same for assessing any task from the training.

**Transaction databases** We parsed the dataset of commands to create two transaction databases used as input for ARM. The *command transaction database* represents each submitted command as a separate transaction, and its goal is to reveal different properties of command usage. Each transaction contains four items that represent the attributes of the command:

- `tool`, the name of the submitted command (e.g., `nmap` or `ssh`),
- `args`, the command-line arguments supplied to the tool,
- `app`, either Bash shell (Linux terminal) or Metasploit,
- `gap`, the time difference between the current and the following command.

For example, the command from Fig. 4 can become a single transaction `{tool = nmap, args = --help, app = bash, gap = low}`. To achieve better interpretability, the `gap` attribute was automatically discretized (Romero et al. (2010) ,p. 102): divided into categorical classes from the set {low, medium, high, undefined}, since the exact value in seconds is not too important. We followed the method previously published by McCall and Kölling (2019). First, the `gap` value in seconds was computed for each command. Then, gaps exceeding the arbitrary maximum of 20 minutes were discretized to "undefined". This resolved the cases of long periods of trainee inactivity. The interval cut-off points for "low", "medium", and "high" categories were computed based on the mean gap from all gaps not exceeding the maximum.

The second database, called the *tool transaction database*, contains transactions with only two attributes: `tool` and `gap`. We merged the consecutive uses of the same tool (regardless of the arguments) into a single transaction. The `gap` represents the time difference between the first use of a tool and the next use of a different tool; the values were discretized as before. The motivation for creating this database was to determine the difficulty of using different tools. If a tool is associated with long gaps, it may indicate that the trainees were unfamiliar with this tool and had difficulties using it.

**Sequence databases** Three sequence databases were created as input for SPM. All three had 113 sequences (corresponding to the number of trainees and the command log files), differing only in the contained items.

The first database, called *command sequence database*, consists of sequences of executed commands. Each item represents a single command, both the tool and its

arguments. For example, a sequence from this database can look like this: `nmap --help, nmap 1.2.3.4, nmap -p 1000 1.2.3.4`.

The second database, *tool sequence database*, contains sequences of tools only. Data from both Bash and Metasploit applications are included in the first two databases. This allows discovering longer patterns, which more accurately reflect the trainees' progress.

The third database, *application sequence database*, stores sequences of applications utilized by the trainees to execute commands. Its goal is to reveal a high-level overview of alternating between applications. This database contains only two unique items: `terminal`, which includes all the commands executed in the Bash shell, and `metasploit`. Table 1 shows the number of transactions/sequences and unique items in each of our databases.

**Association rule mining** For ARM, we used Apyori (Mochizuki, 2019), the Python implementation of the Apriori algorithm. The `MinSup` threshold was manually tuned for each database since there is no simple method to determine it. The threshold was initially set to higher values and then gradually lowered to 0.01–0.04 until we reached a sufficient number of patterns manageable for interpretation. This approach is suggested by Fournier-Viger (2013a) since finding suitable values depends on the data and specific use case.

The `MinConf` threshold is generally easier to set, because the database's properties influence `MinSup` more heavily than `MinConf` (Fournier-Viger et al., 2012). Since we were interested in rules with higher confidence, we used higher `MinConf` thresholds of 0.5. In contrast, `MinSup` needed to be much lower to extract a sufficient amount of rules. This was probably because our transaction databases contained many unique items relative to the total amount of transactions. If there were fewer unique items, `MinSup` could have been increased.

**Sequential pattern mining** For SPM, we used an open-source data mining library SPMF (Fournier-Viger et al., 2016). It provides optimized and documented implementations of more than 190 data mining algorithms (Fournier-Viger, 2021b) often used as benchmarks in research papers (Fournier-Viger et al., 2016). We selected CloFast (Fumarola et al., 2016), an efficient algorithm for mining closed sequential patterns. The `MinSup` threshold was experimentally set from 0.3 to 0.7.

**Table 1** The number of transactions or sequences and unique items contained in each database (DB) for pattern mining, separated for both Training A and B

| Database | Transactions or sequences (Training A / B) | Unique items (Training A / B) |
|---|---|---|
| Command transaction DB | 5700 / 3134 | 1932 / 1092 |
| Tool transaction DB | 4167 / 2062 | 369 / 155 |
| Command sequence DB | 72 / 41 | 2076 / 1155 |
| Tool sequence DB | 72 / 41 | 365 / 151 |
| Application sequence DB | 72 / 41 | 2 / 1 |

### 4.4 Clustering

A popular density-based algorithm is *OPTICS* (Ordering Points To Identify the Clustering Structure) (Ankerst et al., 1999), an improved extension of a widely-used DBSCAN algorithm (Tang et al., 2016). For a data point to belong in a cluster, it must have at least `MinPts` points within its radius.

The result of OPTICS clustering is a *reachability plot*. On the x-axis, it sorts all data points in the order of processing based on their similarity. Values on the y-axis represent the distance of a point from a previous one. Several similar points form a valley representing a cluster, while spikes represent noise or outliers (Ankerst et al., 1999).

In our research, we first represented each command as a Python object with the following attributes: *tool*, *arguments*, *application type*, and *timestamp*, simplifying the record in Fig. 4. Then, we used the commands in two different feature matrices that later act as an input for clustering.

**Bag of words matrix** *Bag of words* model is a standard technique for obtaining features from text (Pelánek et al., 2018). Each text document is represented by a set of words it contains and their count. In our case, the "document" is a command history, and each tool is a "word". We disregarded the command's arguments since we would obtain too many unique features and impair the performance of the clustering algorithm.

**Matrix of selected features** While the *bag of words* model captures the used commands, it does not consider other information available in the logs. Therefore, we selected five custom features to capture other insights into how the trainees progressed:

- `bash-count`, the number of submitted Bash commands. A small number may suggest that a trainee did not progress far in training. The high number may indicate using a trial and error approach.
- `msf-count`, the number of Metasploit commands a trainee used. Metasploit may be new for some trainees, and the high number of executed commands may indicate difficulties with this part of the training.
- `avg-gap`, the average delay between two commands. Large gaps between commands may suggest the trainee did not understand how to use a tool and possibly looked for the information online. Small delays may indicate brute-force guessing.
- `opt-changes`, the number of times trainee used the same tool twice in a row but changed the options or arguments. A high count may show the trainee's unfamiliarity with the tool or inability to use it.
- `help-count`, the number of times trainee displayed help information or manual page for any tool. It may also indicate the trainees' unfamiliarity with the tool.

All features were standardized, namely scaled by their maximum absolute value (scikit-learn developers, 2021). We also checked the Pearson correlation between features, as a high value may make them redundant. While there was a correlation of 0.85 between `bash-count` and `opt-changes`, we preserved both because they capture different properties. All other features were correlated less (the absolute values ranged from 0.20 to 0.66).

**Clustering analysis** We chose the OPTICS algorithm to cluster our data. For calculating the distance between data points, we selected cosine similarity. This measure performs well on high-dimensional data and is often used to compute text similarity (Shirkhorshidi et al., 2015). For example, the command `nmap -sn -PS22 10.1.26.9` has the similarity of 0.6 with the command `nmap --script=vuln 10.1.26.9` and approx. 0.32 with the command `nmap --help`.

During the setup, OPTICS takes only one parameter `MinPts`: the minimum number of points required for cluster formation. Theory suggests setting the number to $ln(n)$, where $n$ is the number of points in the dataset (Birant and Kut, 2007). For our dataset, the recommended value should be close to $ln(113) \approx 5$, which we selected.

## 5 Results and discussion

This section answers the two research questions (RQ) about insights gathered from pattern mining and clustering. We visualize and interpret the findings from specific training sessions and subsequently compare the two approaches.

### 5.1 RQ1: Pattern mining

We now describe and discuss the results revealed by ARM and SPM.

**Transaction databases** The *command transaction database* revealed 51 association rules for Training A and 50 for Training B. Table 2 presents the selected rules marked as interesting by measures such as lift. The first row shows that in Training A, 64% of commands executed in Metasploit had small gaps (delay times). This can mean that using Metasploit involved a rapid sequence of simple commands, or that the trainees experimented with a trial and error approach. The high support of the rule (23%) can also indicate the overuse of Metasploit because it was needed only for one task in this training.

Generally, tools without arguments were associated with small gaps and often with Bash terminal commands. This most likely implies that tools without arguments are easier and faster to use. On the other hand, if a tool had medium or large gaps, it was used in the Bash terminal as well. This is because Bash offers many tools with various difficulty levels, some of which offer a multitude of options.

**Table 2** Association rules mined from *command transaction database* for Training A (rules A*x*) and Training B (rules B*x*)

| # | Rule | Sup | Conf |
|---|---|---|---|
| A1 | `APP=metasploit` → `GAP=low` | 0.23 | 0.64 |
| A2 | `ARGS=[]` → `APP=terminal` | 0.20 | 0.66 |
| A3 | `ARGS=[]` → `GAP=low` | 0.20 | 0.64 |
| A4 | `ARGS=[] GAP=low` → `APP=terminal` | 0.15 | 0.74 |
| A5 | `ARGS=[] APP=terminal` → `GAP=low` | 0.15 | 0.71 |
| A6 | `GAP=medium` → `APP=terminal` | 0.12 | 0.64 |
| A7 | `GAP=high` → `APP=terminal` | 0.11 | 0.69 |
| B1 | `APP=terminal` → `GAP=low` | 0.64 | 0.64 |
| B2 | `ARGS=[]` → `GAP=low` | 0.21 | 0.69 |

The antecedent and consequent of each rule are separated by "→". *Sup* and *Conf* stand for support and confidence rounded to two decimal places

The *tool transaction database* provides further insight into the tool usage. Tools such as `cd`, `ls`, and `cat`, as well as Metasploit commands (`use`, `set`, `show`) were associated with small gaps. However, `nmap` was associated with large gaps in 72% of cases. This can indicate its difficulty of use or the long duration of the scan, which depends on the used arguments, as previously observed by Weiss et al. (2016).

**Sequence databases** The *command sequence database* in Training A revealed that trainees performed the Metasploit exploitation in various ways. Some steps were optional or performed in arbitrary order. When multiple approaches to a solution are possible, instructors can use this insight to show different examples in class, assess all the correct sequences as passed, or even discover novel solutions. Alternatively, when unsuitable subsequences are found, the trainees can be notified, corrected, or even penalized.

In Training B, SPM showed that most trainees established an SSH connection only on the second or third try. When students learn error-prone actions, instructors should leave room for trial and error and not penalize the students for repeated tries. On the other hand, about a third of the trainees excessively used the `ls` tool (as much as 17 times within a single sequence, interleaved by other tools). Instructors should discourage unproductive behavior and maybe offer hints to students when such sequences are observed.

The patterns from the *tool sequence database* show that in Training A, the participants usually progressed as instructors expected. They started with an `nmap` scan and proceeded with the Metasploit exploitation. This is visualized in Fig. 5 using a *Sankey diagram*. Nodes represent the items of the discovered patterns. Edges between the nodes represent subsequences of the patterns. The thicker the edge, the higher the support of the pattern in which the subsequence occurs.

The canonical solution featured these steps in the following order:

- `nmap <target>` – scan the target IP address to discover available services;

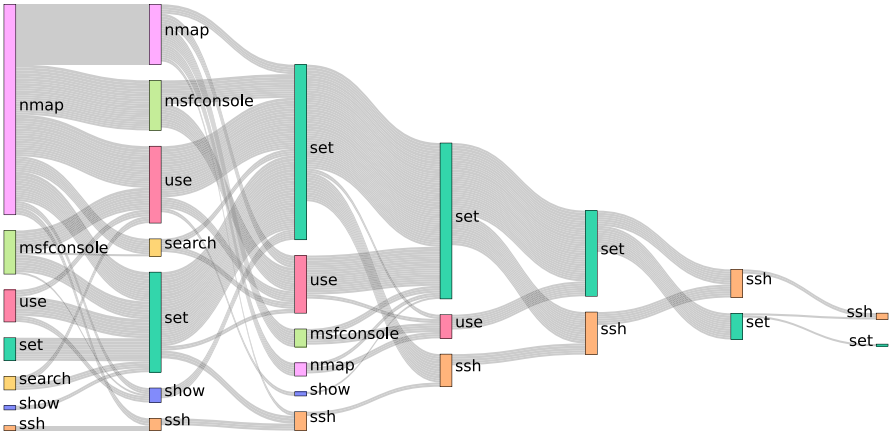


**Fig. 5** A Sankey diagram of closed sequential patterns mined from the *tool sequence database*. The commands `search`, `use`, `show`, and `set` were used in Metasploit; the others in Bash

- `search <keyword>` – find Metasploit exploits suitable for the discovered service based on the provided keyword;
- `use <exploit>` – select the correct exploit;
- `show options` – display parameters of the exploit that need to be set;
- `set <option>` – configure the exploit parameters (used three times to set three mandatory options);
- `run` or `exploit` – execute the exploit script.

Figure 5 shows that most trainees did not use `search`, which suggests they received a hint about which exploit to use. This hint was available in the training platform. Moreover, few of them used `show` to display the exploit options. Instead, they started configuring them immediately, which again suggests they received a hint about which parameters the exploit has and how to configure them. Since the training offered an option to take hints, these actions were legitimate in our context.

In Training B, the longest patterns feature sequences of `ls` and `cd` tools. This can indicate that the trainees struggled to find the files necessary to advance in the task. Again, instructors can provide hints to help the students who become stuck.

Finally, the *application sequence database* confirms our intuition that in Training A, the trainees did not often alternate between Bash and Metasploit. Instead, they used them in longer sequences. For example, the longest discovered pattern features 7 Bash commands, then 8 Metasploit commands, and then 5 Bash commands. The support of this pattern is 0.71, meaning that 71% of trainees behaved this way.

In Training B, the sequence of 12 Bash commands has the support of 0.98, meaning that all but one trainee executed at least 12 commands. If a trainee uses too few commands, it may indicate issues with the assignment, a surprisingly effective solution, outside help, or even cheating.

**Limitations of pattern mining** Setting the `MinSup` and `MinConf` parameters must often be done by trial and error, since there is no universal guide. Also, pattern mining algorithms extracted relatively many patterns, many of which were trivial, for example, `TOOL=cd`→ `APP=terminal`.

Defining the importance of patterns can address this problem. For example, a pattern describing relationships between `tools` would be more important than the usage of the terminal (`app`) itself. Alternatively, additional postprocessing can remove trivial patterns to save the analyst's time. A text-based file defining uninteresting patterns can be used to filter the patterns.

Finally, it can be difficult to interpret why certain patterns occur. Additional information and context are needed to maximize the usefulness of extracted patterns.

**Summary of RQ1** ARM and SPM are suitable for uncovering the following educational insights:

- Approaches to solving the tasks, namely typical associations of tools and their arguments (for ARM) or sequences of commands (for SPM).
- Mistakes and errors based on incorrectly used tools or unknown commands and sequences.
- Problematic tasks within the training, such as when a student attempts to use a tool several times in a row (for SPM).
- Novel solutions, such as when unexpected but correct tools appear in a rule or a sequence.
- Tools used at the beginning or toward the end of the task, based on whether the sequences often begin or end with a certain tool (for SPM).
- Frequency of tools' usage, such as the commands utilized by most trainees or overuse of a tool, which is proportional to the rule's or sequence's support.
- Timing information, namely small or large gaps between two submissions of commands associated with certain tools (for ARM).

The results of pattern mining can be tabulated (see Table 2) or visualized in a Sankey diagram, such as the one in Fig. 5.

### 5.2 RQ2: Clustering

Now, we continue with the results of clustering the *bag of words* and *selected features* matrices.

**Bag of words cluster ordering** When clustering the 72 trainees from Training A, 31 trainees form Cluster 1, 14 trainees constitute Cluster 2, 5 trainees form Cluster 3, and 22 were designated as outliers. Cluster 2 is the most compact because of low reachability distance between the points. This implies that the trainees progressed strongly similarly.

When visualizing the most common combinations of tools and arguments (see Fig. 6), we discovered that Cluster 1 trainees used `nmap` and `ssh` with certain

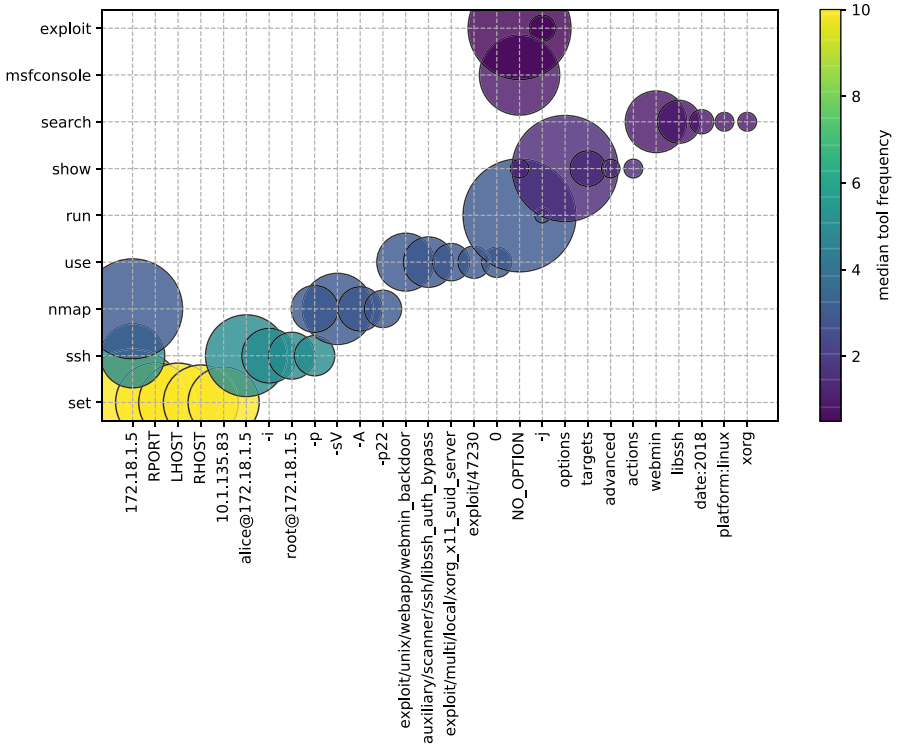


**Fig. 6** Bubble plot showing the most common `tool` and `argument` combination for Cluster 1. The size of the bubble is correlated with the argument frequency. The color represents the median tool frequency for the cluster

arguments slightly more often. For example, they executed a correct `nmap` scan multiple times, maybe to assure themselves of the results.

Trainees from Clusters 1 and 3 also experimented much more with setting the Metasploit exploit options, and attempted to search for and use several different exploits. Cluster 2 trainees selected and configured the suitable exploit on fewer tries. A relatively low number of Metasploit commands and the lack of option variety suggest that Cluster 2 trainees did not struggle with Metasploit. Instructors can use this information to check in with Cluster 1 and 3 trainees and ask them whether they are stuck or need assistance.

Cluster 2 trainees used more Bash commands on average, and they used commands for changing and listing directories (`cd` and `ls`) overwhelmingly more often. They probably had trouble locating the files crucial for the task. Based on this insight, instructors can again provide targeted help to trainees in this cluster.

For the 41 trainees in Training B, the clustering was not too fruitful. 13 trainees formed a cluster, while the remaining 28 were designated as outliers. The trainees in a cluster again used a lot of `cd` and `ls` tools, and experimented with `scp` and `fcrackzip` more often than the remaining trainees.

Examining the trainees designated as outliers can also yield interesting results. One of the outliers did not use `nmap` for network scanning, but `ike-scan` and then `zenmap`. This shows that alternative tools are possible for solving the tasks, and outliers can still be successful in the training. It is worth noting that even small differences or deviations in a single task can be enough for the trainee to be considered an outlier. However, these cases have to be further investigated manually.

Finally, another outlier brute-forced the searching of argument combinations for `john`. Therefore, outliers can also be trainees behaving problematically.

**Selected features cluster ordering** Clustering of Training A data formed four clusters: with 6, 9, 14, and 6 members, respectively. Cluster 1 had the largest count of Metasploit commands and the smallest average time gap between their submitted commands. These differences were also confirmed by pairwise t-tests statistically significant at $p \leq 0.01$. Since only a few Metasploit commands were needed to reach the solution, this indicates a trial-and-error approach of trainees in this cluster. Moreover, these trainees did not display the manual pages or the tools' usage help. Such unproductive behavior can be automatically recognized, along with notifying the instructors. On the other hand, Cluster 4 trainees displayed command help the most often, which can be suitable while learning.

Cluster 2 and Cluster 3 behaved in an almost opposite ways. The former used the most Bash commands with relatively small gaps, and the latter used the least Bash commands with the largest gaps. This suggests that Cluster 3 trainees did not progress far, perhaps due to lack of motivation or skill.

Two clusters emerged in Training B. The first had fewer submitted commands with larger gaps. The second submitted many commands with small gaps and changed the command arguments often. These trainees probably struggled to figure out the correct argument combination.

Overlaps with the *bag of words* clusters were minimal, suggesting that the results largely depend on the chosen features. Both approaches can provide useful insights; however, as in almost all machine learning approaches, it is difficult to select the best features.

**Limitations of clustering** The main limitation of clustering is that determining relevant features is hard. In addition, the relatively small sample size was problematic for the chosen clustering algorithm. Sometimes, only one cluster was formed, or only a few data points belonged to a cluster. Nevertheless, the format of the training implies that massive amounts of command histories cannot be collected.

**Summary of RQ2** Clustering can reveal the following educational insights:

- Similarities and differences between trainees' approaches to the training, for example, in typically used combinations of tools.
- Alternative solutions to training tasks based on examining outliers.
- Behavioral patterns, such as help-seeking or submitting many commands in a rapid succession.

**Table 3** The insights and situations that can arise during the training

| Positive or neutral observations | Negative observations |
|---|---|
| P1 The trainee is proficient | N1 The trainee lacks skill |
| P2 The trainee is motivated | N2 The trainee is demotivated |
| P3 The trainee progresses smoothly | N3 The trainee experiences difficulties |
| P4 The trainee received allowed help | N4 The trainee received prohibited help |
| P5 The trainee corrected a mistake | N5 The trainee uses a trial-and-error approach |
| P6 The trainee discovered a novel solution | |
| P7 The trainee is taking a break | |
| P8 The tool executes quickly | N8 The tool executes slowly |
| P9 The tool is easy to use | N9 The tool is difficult to use |
| | N10 The tool is used too little |
| | N11 The tool is used too much |
| P12 The task features simple commands | |
| | N13 The task is not designed clearly |

**Table 4** The comparison of results of pattern mining (ARM, SPM) and clustering (C)

| Method | Insight category | Result | Explanations |
|---|---|---|---|
| ARM, C | Solution approaches | Typical combinations of tools | task-dependent |
| SPM | Solution approaches | Typical sequences of tools | task-dependent |
| all | Tool use frequency | Low rule support / use count | P6, N1, N10 |
| | | High rule support / use count | N1, N5, N11, N13 |
| ARM, C | Timing information | Low command delay | P1, P4, P8, P9, P12, N4 |
| | | High command delay | P7, N1, N2, N3, N8 |
| C | Trainee similarities | Common behavioral patterns | task-dependent |

The column *Explanations* refers to possible causes in Table 3

The results of clustering and the associated features can be easily visualized or tabulated, which provides a straightforward overview (see, e.g., Fig. 6).

### 5.3 Comparison of the two approaches

We used two approaches, *pattern mining* and *clustering*, to analyze data from cybersecurity assignments completed via a command line. Table 3 summarizes the different situations that can occur during the training and are of interest to instructors. Then, Table 4 provides a grand overview of insights discoverable with the two approaches. Not all of them were demonstrated by our data, but they can be investigated by future research.

These insights can also occur in combination. For example, a low frequency of command usage combined with large gaps probably suggests demotivation or lack of skill.

**Similarities of pattern mining and clustering** Both pattern mining and clustering can reveal trainees' strategies utilized to solve the tasks. These include desirable solutions, mistakes and errors, and novel approaches. They can also highlight statistical properties of the solutions, such as frequently used tools or their time gaps.

Methodically, the process of pattern mining and clustering is relatively straightforward. As long as the input data format and constraints are preserved, it is sufficient to use existing implementations of these algorithms.

Insights from both pattern mining and clustering can be targeted at specific trainees. Instructors can provide suitable feedback to the whole cluster of students or all students whose logs matched a specific pattern. This way, instructors can help the struggling trainees if they exhibit signs typical for low-performing clusters or associated with undesirable patterns.

**Differences of pattern mining and clustering** Setting the initial parameters appears to be easier for the OPTICS clustering algorithm compared to ARM and SPM. OPTICS recommends setting `MinPts` approximately to the natural logarithm of the sample size. For pattern mining, the `MinSup` and `MinConf` parameters need to be set experimentally. Nevertheless, clustering requires a careful selection of features, which can include the collected data as well as properties derived from them.

Density-based clustering is more prone to small sample size. On the contrary, our dataset of thousands of commands was sufficient for pattern mining. In fact, most public datasets previously used for ARM contain thousands up to a million transactions, and datasets for SPM start with as little as ten sequences up to ten thousand (Fournier-Viger, 2021a). (However, these datasets come from other domains, such as word corpora or clickstream data from websites.) As a result, educational researchers who have just begun to collect data may experience the cold start problem. Especially when using clustering, their early dataset will not be large enough to provide insights about the first few students.

Finally, the results of clustering are more easily interpretable. Pattern mining can yield a large number of trivial patterns. Nevertheless, interpreting the clustering results requires further investigation of the properties of the discovered clusters. Patterns are readable directly.

### 5.4 Comparison with related work

Section 3 reviewed numerous approaches to student assessment and usage of pattern mining and clustering to analyze educational data. We now compare our methods and results with those presented in related work in order to highlight our contributions.

One novel aspect of our research is the application domain of cybersecurity, since most of the related work focused on other areas, such as programming education (Gao et al., 2021; Yin et al., 2015; McBroom et al., 2016; Piech et al., 2012; Emerson et al., 2020; Wiggins et al., 2021). Educational researchers and practitioners in

cybersecurity and related domains, such as operating systems and networking, may benefit from the presented evaluation featuring authentic cybersecurity training data.

Several learning environments for cybersecurity allow logging command-line interactions (Švábenský et al., 2021; Mirkovic et al., 2020; Andreolini et al., 2019; Labuschagne and Grobler, 2017; Tian et al., 2018), although this practice is still relatively rare. Nevertheless, even if interesting datasets are acquired, few methods have been explored for their automated analysis (see Section 3.4).

Student assessment in cybersecurity was specifically reviewed in Section 3.3. Based on inspecting the related work, we believe this is the first study evaluating the applicability of pattern mining and clustering algorithms on cybersecurity training data. Other works focused, for example, on generating progress models of students (Švábenský et al., 2022), predicting their success (Vinlove et al., 2020), or assessing team performance (Granåsen & Andersson, 2016; Henshel et al., 2016; Maennel et al., 2017).

### 5.5 Educational implications

Our research demonstrated the automation of discovering educational insights. Previously, these insights had to be revealed manually by the instructor, which was time-consuming, or were even completely unavailable. In particular, the insights gained from pattern mining and clustering have the following implications for teachers, educational researchers, and other stakeholders:

- *Classroom-wide instruction* – instructors can show the typical or rare solution approaches to the students and discuss them together. They can also explain the erroneous or novel solutions. If students are aware of examples of good or bad practices, they can follow or avoid them, respectively.

  For example, in our data, `nmap` was often associated with high time gaps. Instructors can revisit the explanation of this tool and stress how its argument combinations affect the scan duration.
- *Targeted instruction* – when a student exhibits patterns associated with errors or unproductive behavior, the instructor can intervene appropriately. This intervention can include providing tailored hints, feedback, scaffolding, or suitably correcting the student. Identifying struggling students early and helping them is crucial for supporting their learning.

  For example, we discovered a cluster of trainees who adopted a trial-and-error approach when using Metasploit. If this happened during class, instructors could visit these students in real-time and provide suitable assistance. By identifying specific students belonging to the cluster, the instructor can save time by providing the same help to all students in that cluster.
- *Marking/grading* – knowing the common mistakes aids with both manual and automated grading. Instructors can create a grading rubric based on the observed errors and approaches. Moreover, an autograder can be set up to grade specific actions as passed or failed.

For example, the trainees who used a correct sequence of Metasploit commands to configure all exploit steps can be awarded a point.

- *Task design* – based on summarizing the common approaches of students and discovering novel approaches to the solution, instructors can design more suitable tasks. This includes fixing unclear or problematic tasks.

  For example, we discovered that in Training B, participants excessively used the `cd` and `ls` tools to traverse the filesystem. The assignment can specify more clearly what type of file to look for and where. It can also feature hints.
- *Machine learning* – the patterns and their features (such as using a particular command or making a specific mistake) can act as input in other machine learning models for further analysis and student modeling.

  For example, the discovered patterns could be used to train a classifier to predict student success or failure.
- *Curricular support* – several policies and curricular guidelines in cybersecurity (CC2020 Task Force, 2020; Joint Task Force on Cybersecurity Education, 2017; Parrish et al., 2018) prescribe *what* skills should be taught and assessed. However, the information about *how* to perform this assessment is left to the educators. Our paper demonstrates a possible solution for assessing hands-on exercises in cybersecurity, which other educators can adopt or adapt.

## 6 Conclusion

Automated student assessment in cybersecurity is becoming more and more relevant. Finding better ways to analyze data from cybersecurity training is needed to support more effective hands-on training. Yet, this research area is still in its early stages.

To contribute to the body of knowledge on student assessment, we investigated automated methods for analyzing log data from authentic educational contexts. We mined 8834 commands from several-hours-long training sessions with small groups of computing students. Then, we discussed the observations relevant for instructors and researchers in cybersecurity and beyond.

Our results include the prototype implementation, evaluation, and comparison of two data mining approaches within specific cybersecurity training, as well as general insights and lessons learned. Answering our two research questions revealed that:

1. *Pattern mining* is suitable for revealing solution approaches of students, their misconceptions, and difficult training tasks.
2. *Clustering* highlights similarities and differences between approaches of students, grouping them based on their behavioral patterns.

Other educators can use these insights to improve cybersecurity training in their context or adapt them to training in other domains. Pattern mining and clustering are suitable for any problem-solving assignments that yield interaction data. Instructors can exploit these data to identify and redesign problematic sections of the training,

reveal new solutions to the tasks, and provide targeted instruction and feedback to trainees.

### 6.1 Practical contributions and supplementary materials

In addition to educational implications described in Section 5.5, we share numerous artifacts with the community of instructors, researchers, and developers. These artifacts enable replicating our study setup and advancing the research in cybersecurity education. Moreover, the tools are applicable for hands-on security classes. These artifacts are open-source and include:

- Cybersecurity training content (Masaryk University, 2022b), which can be deployed in either of our learning environments: KYPO Cyber Range Platform (Masaryk University, 2021) and Cyber Sandbox Creator (Masaryk University, 2022a). Cybersecurity instructors can freely use them to host cybersecurity training sessions (Vykopal et al., 2021).
- Logging infrastructure (Švábenský et al., 2021), which enables researchers to collect command-line data like for this paper.
- The analyzed dataset, which has been published with records from other trainings (Švábenský et al., 2021). Since each record is a command submitted by a person, accumulating these data is a challenge on its own. Therefore, such datasets are rare and may help other researchers.
- The created software that applies pattern mining and clustering on the data. It includes a Python implementation of extracting and visualizing the patterns and clusters. This implementation can serve as a starting point for the developers of learning environments when integrating the researched methods (Švábenský et al., 2022).
- Visualizations and the full results (Švábenský et al., 2022).

### 6.2 Future work

This research offers many possibilities for extension. In pattern mining, transaction databases can include time-related information, such as the duration of running a command. This would distinguish a difficult task from a command that took a long time to execute. Sequential databases can include timestamps to describe time gaps between sequences in a pattern. Additionally, the dataset can be expanded with information about other actions of trainees, such as asking for a hint. As a result, we would discover sequences that preceded help-seeking. Finally, we can consider the whole command history of a student as a single transaction to generate new types of insights.

Enhancements are possible for clustering as well. One is the clustering of time series: each training would be represented as time series of commands encoded as vectors. The other option is to use different algorithms, such as hierarchical clustering.

Yet another extension is to incorporate live data mining during an ongoing training. Online algorithms can provide relevant insights to instructors in real-time. Their results could also be used as a basis for a recommender system that would provide hints for stuck trainees. If a trainee needs help, the system could recommend a hint that helped another trainee from the same cluster.

**Acknowledgements** We thank Radek Pelánek and Tomáš Effenberger for their valuable feedback that helped with the early stages of this paper. We also thank the anonymous reviewers who provided useful perspective to improve the final version of the paper.

**Author contributions** *Valdemar Švábenský*: Conceptualization, Methodology, Validation, Investigation, Resources, Data Curation, Writing – Original Draft, Visualization, Supervision, Project administration. *Jan Vykopal*: Investigation, Resources, Writing – Review & Editing. *Pavel Čeleda*: Writing – Review & Editing, Funding acquisition. *Kristián Tkáčik*: Software, Formal analysis, Data Curation, Writing – Review & Editing, Visualization. *Daniel Popovič*: Software, Formal analysis, Writing – Review & Editing, Visualization.

**Funding** This research was supported by the ERDF project *CyberSecurity, CyberCrime and Critical Information Infrastructures Center of Excellence* (No. CZ.02.1.01/0.0/0.0/16_019/0000822).

**Availability of data and materials** The accompanying software code and results are published in a Zenodo repository (Švábenský et al., 2022). The dataset was published in a separate data article (Švábenský et al., 2021) (this paper used a subset of the published data).

## Declarations

**Conflict of interests** The authors have no competing interests to declare.

## References

Abbott, R.G., McClain J., Anderson, B., Nauer, K., Silva, A., & Forsythe, C. (2015). Log analysis of cyber security training exercises. *Procedia Manufacturing*, *3*, 5088–5094. https://doi.org/10.1016/j.promfg.2015.07.523

Aggarwal, C.C. , Hinneburg A., & Keim D.A. (2001). On the surprising behavior of distance metrics in high dimensional space. In J. Van den Bussche V. Vianu (Eds.) *Database Theory — ICDT 2001*. https://doi.org/10.1007/3-540-44503-X_27 (pp. 420–434). Berlin, Heidelberg: Springer.

Andreolini, M., Colacino, V.G., Colajanni, M., & Marchetti, M. (2019). A framework for the evaluation of trainee performance in cyber range exercises. *Mobile Networks and Applications*, *25*, 236–247. https://doi.org/10.1007/s11036-019-01442-0

Ankerst, M., Breunig, M.M., Kriegel, H.-P., & Sander, J. (1999). OPTICS: Ordering points to identify the clustering structure. *SIGMOD Record*, *28*(2), 49–60. https://doi.org/10.1145/304181.304187

Beyer, K., Goldstein, J., Ramakrishnan, R., & Shaft, U. (1999). When is "nearest neighbor" meaningful? In C. Beeri P. Buneman (Eds.) *Database Theory — ICDT'99*. https://doi.org/10.1007/3-540-49257-7_15 (pp. 217–235). Berlin, Heidelberg: Springer.

Bienkowski, M., Feng, M., & Means, B. (2012). Enhancing teaching and learning through educational data mining and learning analytics: An issue brief. *US Department of Education, Office of Educational Technology*, 1, 1–57. https://files.eric.ed.gov/fulltext/ED611199.pdf

Birant, D., & Kut, A. (2007). ST-DBSCAN: An algorithm for clustering spatial–temporal data. *Data & Knowledge Engineering*, *6*(1), 208–221. https://doi.org/10.1016/j.datak.2006.01.013

CC2020 Task Force. (2020). *Computing Curricula 2020: Paradigms for global computing education*. New York NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3467967

Dutt, A., Ismail, M.A., & Herawan, T. (2017). A systematic review on educational data mining. *IEEE Access*, *5*, 15991–16005. https://doi.org/10.1109/ACCESS.2017.2654247

Emerson, A., Smith, A., Rodriguez, F.J., Wiebe, E.N., Mott, B.W., Boyer, K.E., & Lester, J.C. (2020). Cluster-based analysis of novice coding misconceptions in block-based programming. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. https://doi.org/10.1145/3328778.3366924 (pp. 825–831). New York, NY, USA: Association for Computing Machinery.

Fournier-Viger, P. (2013a). How to auto-adjust the minimum support threshold according to the data size. Retrieved February 9, 2022 from http://data-mining.philippe-fournier-viger.com/how-to-auto-adjust-the-minimum-support-threshold-according-to-the-data-size/

Fournier-Viger, P. (2013b). An introduction to frequent pattern mining. Retrieved February 9, 2022 from http://data-mining.philippe-fournier-viger.com/introduction-frequent-pattern-mining/

Fournier-Viger, P. (2017). An introduction to data mining. Retrieved February 9, 2022 from http://data-mining.philippe-fournier-viger.com/introduction-data-mining/

Fournier-Viger, P. (2021a). Datasets. Retrieved February 9, 2022 from https://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php

Fournier-Viger, P. (2021b). SPMF: An open-source data mining library. Retrieved February 9, 2022 from https://www.philippe-fournier-viger.com/spmf/

Fournier-Viger, P., Gomariz, A., Campos, M., & Thomas, R. (2014). Fast vertical mining of sequential patterns using co-occurrence information. In *Advances in knowledge discovery and data mining*. https://doi.org/10.1007/978-3-319-06608-0_4(pp. 40–52). Springer International Publishing.

Fournier-Viger, P. , Lin, J.C- W., Gomariz, A. , Gueniche, T. , Soltani, A., Deng, Z., & Lam, H.T. (2016). The SPMF open-source data mining library Version 2. In *Machine learning and knowledge discovery in databases*. https://doi.org/10.1007/978-3-319-46131-1_8 (pp. 36–40). Springer International Publishing.

Fournier-Viger, P., Lin, J.C.-W., Kiran, R.U., Koh, Y.S., & Thomas, R. (2017). A survey of sequential pattern mining. *Data Science and Pattern Recognition*, *1*(1), 54–77.

Fournier-Viger, P. , Wu, C-W., & Tseng, V.S. (2012). Mining Top-K association rules. In *Advances in artificial intelligence*. https://doi.org/10.1007/978-3-642-30353-1_6 (pp. 61–73). Springer Berlin Heidelberg.

Fumarola, F., Lanotte, P.F., Ceci, M., & Malerba, D. (2016). CloFAST: Closed sequential pattern mining using sparse and vertical Id-Lists. *Knowledge and Information Systems*, *48*(2), 429–463. https://doi.org/10.1007/s10115-015-0884-x

Gao, G., Marwan, S., & Price, T.W. (2021). Early performance prediction using interpretable patterns in programming process data. In *Proceedings of the 52nd ACM technical symposium on computer science education*. https://doi.org/10.1145/3408877.3432439 (pp. 342–348). New York NY, USA: Association for Computing Machinery.

García, E., Romero, C., Ventura, S., de Castro, C., & Calders, T. (2010). Association rule mining in learning management systems. In C. Romero, S. Ventura, M. Pechenizkiy, & R.S. Baker (Eds.) *Handbook of educational data mining*. https://doi.org/10.1201/b10274 (pp. 93–103). Boca Raton, FL, USA: CRC Press.

Granåsen, M., & Andersson, D. (2016). Measuring team effectiveness in cyber-defense exercises: a cross-disciplinary case study. *Cognition, Technology & Work*, *18*(1), 121–143. https://doi.org/10.1007/s10111-015-0350-2

Han, J., Kamber, M., & Pei, J. (2011). *Data mining: Concepts and techniques*, 3rd edn. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. https://www.sciencedirect.com/book/9780123814791/data-mining-concepts-and-techniques

Henshel, D.S., Deckard, G.M., Lufkin, B., Buchler, N., Hoffman, B., Rajivan, P., & Collman, S. (2016). Predicting proficiency in cyber defense team exercises. In *MILCOM 2016 – IEEE military communications conference*. https://doi.org/10.1109/MILCOM.2016.7795423 (pp. 776–781). New York, NY, USA: IEEE.

$(ISC)^2$. (2021). Cybersecurity workforce study (Tech. Rep.) https://www.isc2.org/Research/Workforce-Study

Joint Task Force on Cybersecurity Education. (2017). Cybersecurity curricular guideline. Retrieved February 9, 2022 from http://cybered.acm.org/

Kobayashi, Y. (2014). Computer-aided error analysis of L2 spoken English: A data mining approach. *Proceedings of the Conference on Language and Technology*, *2014*, 127–134.

Labuschagne, W.A., & Grobler, M. (2017). Developing a capability to classify technical skill levels within a cyber range. In *16th European conference on cyber warfare and security, ECCWS 2017*. https://www.proquest.com/docview/1966803837(pp. 224–234). Red Hook NY, USA: Curran Associates Inc.

Lancaster, T., Robins, A.V., & Fincher, S.A. (2019). Assessment and plagiarism. In S. A. Fincher A. V. Robins (Eds.) *The cambridge handbook of computing education research*. https://doi.org/10.1017/9781108654555.015 (pp. 414–444). Cambridge, United Kingdom: Cambridge University Press.

Lang, C., Wise, A., & Gašević, D. (Eds). (2017). Handbook of learning analytics (1st ed) Society for Learning Analytics Research (SoLAR). https://doi.org/10.18608/hla17

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, *28*(2), 129–137. https://doi.org/10.1109/TIT.1982.1056489

Madhulatha, T.S. (2012). An overview on clustering methods. *IOSR Journal of Engineering*, *2*(4), 719–725. https://doi.org/10.9790/3021-0204719725

Maennel, K. (2020). Learning analytics perspective: Evidencing learning from digital datasets in cybersecurity exercises. In *2020 IEEE european symposium on security and privacy workshops (EuroSPW)*. https://doi.org/10.1109/EuroSPW51379.2020.00013 (pp. 27–36).

Maennel, K., Ottis, R., & Maennel, O. (2017). Improving and measuring learning effectiveness at cyber defense exercises. In *22nd nordic conference on secure IT systems, NordSec 2017*. https://doi.org/10.1007/978-3-319-70290-2_8 (pp. 123–138). Vienna, Austria: Springer.

Malekian, D., Bailey, J., & Kennedy, G. (2020). Prediction of students' assessment readiness in online learning environments: The sequence matters. In *Proceedings of the tenth international conference on learning analytics & knowledge*. https://doi.org/10.1145/3375462.3375468 (pp. 382–391). New York NY, USA: Association for Computing Machinery.

Masaryk University. (2021). KYPO cyber range platform. Retrieved February 9, 2022 from https://crp.kypo.muni.cz

Masaryk University. (2022a). Cyber sandbox creator. Retrieved February 9, 2022 from https://gitlab.ics.muni.cz/muni-kypo-csc/cyber-sandbox-creator

Masaryk University. (2022b). The listing of all cybersecurity games and common instructions. Retrieved February 9, 2022 from https://gitlab.ics.muni.cz/muni-kypo-trainings/games/all-games-index

McBroom, J., Jeffries, B., Koprinska, I., & Yacef, K. (2016). Mining behaviours of students in autograding submission system logs. In *Proceedings of the 9th international conference on educational data mining*. https://www.educationaldatamining.org/EDM2016/proceedings/paper_172.pdf

McCall, D., & Kölling, M. (2019). A new look at novice programmer errors. *ACM Transactions on Computing Education*, *19*(4), 38:1–38:30. https://doi.org/10.1145/3335814

McClain, J., Silva, A., Emmanuel, G., Anderson, B., Nauer, K., Abbott, R., & Forsythe, C. (2015). Human performance factors in cyber security forensic analysis. *Procedia Manufacturing*, *3*, 5301–5307. https://doi.org/10.1016/j.promfg.2015.07.621

Mirkovic, J., Aggarwal, A., Weinman, D., Lepe, P., Mache, J., & Weiss, R. (2020). Using terminal histories to monitor student progress on hands-on exercises. In *Proceedings of the 51st ACM technical symposium on computer science education*. https://doi.org/10.1145/3328778.3366935 (pp. 866–872). New York NY, USA: Association for Computing Machinery.

Mochizuki, Y. (2019). Apyori. Retrieved February 9, 2022 from https://github.com/ymoch/apyori/

Offensive Security. (2022a). Kali Linux. Retrieved February 9, 2022 from https://www.kali.org/

Offensive Security. (2022b). Metasploit unleashed. Retrieved February 9, 2022 from https://www.offensive-security.com/metasploit-unleashed/

Palmer, N. (2019). Automating the assessment of network security in higher education. In *2019 international conference on computing, electronics communications engineering (iCCECE)*. https://doi.org/10.1109/iCCECE46942.2019.8941804 (pp. 141–146).

Parrish, A., Impagliazzo, J., Raj, R.K., Santos, H., Asghar, M.R., Jøsang, A., & Stavrou, E. (2018). Global perspectives on cybersecurity education for 2030: A case for a meta-discipline. In *Proceedings companion of the 23rd annual ACM conference on innovation and technology in computer science education*. https://doi.org/10.1145/3293881.3295778 (pp. 36–54). New York, NY, USA: ACM.

Pelánek, R., Effenberger, T., Vaněk, M., Sassmann, V., & Gmiterko, D. (2018). Measuring item similarity in introductory programming. In *Proceedings of the fifth annual ACM conference on*

*learning at scale*. https://doi.org/10.1145/3231644.3231676 (pp. 1–4). New York NY, USA: Association for Computing Machinery.

Piech, C., Sahami, M., Koller, D., Cooper, S., & Blikstein, P. (2012). Modeling how students learn to program. In *Proceedings of the 43rd ACM technical symposium on computer science education*. https://doi.org/10.1145/2157136.2157182(pp. 153–160). New York NY, USA: Association for Computing Machinery.

Popovič, D. (2021). Clustering of command histories from cybersecurity training (Bachelor thesis, Masaryk University, Faculty of Informatics. https://is.muni.cz/th/fefjq/?lang=en

Romero, C., & Ventura, S. (2010). Educational data mining: A review of the state of the art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *40*(6), 601–618. https://doi.org/10.1109/TSMCC.2010.2053532

Romero, C., & Ventura, S. (2020). Educational data mining and learning analytics: An updated survey. *WIREs Data Mining and Knowledge Discovery*, *10* (3), e1355. https://doi.org/10.1002/widm.1355

In C. Romero, S. Ventura, M. Pechenizkiy, & R.S. Baker (Eds.) (2010). *Handbook of educational data mining*. Boca Raton, FL, USA: CRC Press. https://doi.org/10.1201/b10274

scikit-learn developers. (2021). sklearn.preprocessing.MaxAbsScaler. Retrieved February 9, 2022 from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MaxAbsScaler.htmlx

Shirkhorshidi, A.S. , Aghabozorgi S., & Wah T.Y. (2015). A Comparison study on similarity and dissimilarity measures in clustering continuous data. *PLoS ONE*, *10*(12), 1–20. https://doi.org/10.1371/journal.pone.0144059

Tang, W., Pi, D., & He, Y. (2016). A density-based clustering algorithm with sampling for travel behavior analysis. In H. Yin et al (Eds.) *Intelligent data engineering and automated learning – IDEAL 2016*. https://doi.org/10.1007/978-3-319-46257-8_25 (pp. 231–239). Cham: Springer International Publishing.

Tian, Z., Cui, Y., An, L., Su, S., Yin, X., Yin, L., & Cui, X. (2018). A real-time correlation of host-level events in cyber range service for smart campus. *IEEE Access*, *6*, 35355–35364. https://doi.org/10.1109/ACCESS.2018.2846590

Tkáčik, K. (2020). Pattern mining in command histories from cybersecurity training (Bachelor thesis, Masaryk University, Faculty of Informatics). https://is.muni.cz/th/cxvr2/?lang=en

Vellido, A., Castro, F., & Nebot, A. (2010). Clustering educational data. In C. Romero, S. Ventura, M Pechenizkiy, & R.S. Baker (Eds.) *Handbook of educational data mining*. https://doi.org/10.1201/b10274 (pp. 75–92). Boca Raton FL, USA: CRC Press.

Vinlove, Q., Mache, J., & Weiss, R. (2020). Predicting student success in cybersecurity exercises with a support vector classifier. *Journal of Computing Sciences in Colleges*, *36*(1), 26–34. https://doi.org/10.5555/3447051.3447055

Švábenský, V., Vykopal, J., Seda, P., & Čeleda, P. (2021). Dataset of shell commands used by participants of hands-on cybersecurity training. *Data in Brief*. https://doi.org/10.1016/j.dib.2021.107398

Švábenský, V., Vykopal, J., Tovarňák, D., & Čeleda, P. (2021). Toolset for collecting shell commands and its application in hands-on cybersecurity training. In *Proceedings of the 51st IEEE frontiers in education conference*. https://doi.org/10.1109/FIE49875.2021.9637052 (pp. 1–9). New York NY, USA: IEEE.

Švábenský, V., Vykopal, J., Čeleda, P., Tkáčik, K., & Popovič, D. (2022). *Supplementary Materials: Student assessment in cybersecurity training automated by pattern mining and clustering*. Zenodo. https://doi.org/10.5281/zenodo.6024825

Švábenský, V., Weiss, R., Cook, J., Vykopal, J., Čeleda, P., Mache, J., & Chattopadhyay, A. (2022). Evaluating two approaches to assessing student progress in cybersecurity exercises. In *Proceedings of the 53rd ACM technical symposium on computer science education*. https://doi.org/10.1145/3478431.3499414. New York NY, USA: Association for Computing Machinery.

Vykopal, J., Čeleda, P., Seda, P., Švábenský, V., & Tovarňák, D. (2021). Scalable learning environments for teaching cybersecurity hands-on. In *Proceedings of the 51st IEEE frontiers in education conference*. https://doi.org/10.1109/FIE49875.2021.9637180 (pp. 1–9). New York, NY, USA: IEEE.

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

*science education*. https://doi.org/10.1145/2839509.2844646 (pp. 597–602). New York, NY, USA: ACM.

Weiss, R., Turbak, F., Mache, J., & Locasto, M.E. (2017). Cybersecurity education and assessment in EDURange. *IEEE Security & Privacy*, *15* (3), 90–95. https://doi.org/10.1109/MSP.2017.54

Wiggins, J.B., Fahid, F.M., Emerson, A., Hinckle, M., Smith, A., Boyer, K.E., & Lester, J. (2021). Exploring novice programmers' hint requests in an intelligent block-based coding environment. In *Proceedings of the 52nd ACM technical symposium on computer science education*. https://doi.org/10.1145/3408877.3432538(pp. 52–58). New York, NY, USA: Association for Computing Machinery.

Yin, H., Moghadam, J., & Fox, A. (2015). Clustering student programming assignments to multiply instructor leverage. In *Proceedings of the second (2015) ACM conference on learning at scale*. https://doi.org/10.1145/2724660.2728695 (pp. 367–372). New York, NY, USA: Association for Computing Machinery.



## Authors and Affiliations

**Valdemar Švábenský[1,2] · Jan Vykopal[1] · Pavel Čeleda[1] · Kristián Tkáčik[2] · Daniel Popovič[2]**

Jan Vykopal
vykopal@ics.muni.cz

Pavel Čeleda
celeda@ics.muni.cz

Kristián Tkáčik
tkacikk@mail.muni.cz

Daniel Popovič
popovic@mail.muni.cz

[1] Institute of Computer Science, Masaryk University, Šumavská 15, Brno 60200, Czech Republic

[2] Faculty of Informatics, Masaryk University, Botanická 68a, Brno 60200, Czech Republic