Masaryk University
Faculty of Informatics

Pattern Mining in Command Histories from Cybersecurity Training

Bachelor's Thesis

Kristián Tkáčik

Brno, Spring 2020

This is where a copy of the official signed thesis assignment and a copy of the Statement of an Author is located in the printed version of the document.

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Kristián Tkáčik

Advisor: RNDr. Valdemar Švábenský

Acknowledgements

I would like to thank my advisor RNDr. Valdemar Švábenský for his guidance. His frequent feedback, invaluable advice, and immense time investment were essential to creating this thesis. His friendly, supportive, and responsible approach have made the entire process a very pleasant experience.

My gratitude also goes to my family and friends, whose support provided me with the confidence and strength to finish this thesis.

Abstract

Modern educational environments for practicing cybersecurity allow instructors to collect command histories from cybersecurity training. Analysis of these data can uncover valuable insights about the trainees' learning processes. However, manual transformation of the data into useful information is time-consuming and ineffective. In this thesis, we applied *association rule mining* and *sequential pattern mining* algorithms to command histories from 22 runs of cybersecurity training. Our aim was to explore interesting patterns in the collected data, visualize and analyze them, and interpret them into useful educational insights. The results provided information concerning common mistakes, misconceptions, and strategies the trainees utilized, as well as difficult concepts and stages of the training. The results proved that pattern mining methods are a viable option for analyzing data originating from cybersecurity training. Instructors can apply these methods in their contexts and use the gained insights to improve cybersecurity education.

Keywords

data analysis, security, cybersecurity training, Python, KYPO, education, data mining, pattern mining, association rule mining, sequential pattern mining

Contents

1 Introduction

There is a rising need for cybersecurity experts. It was estimated that by the end of 2019, there were 4.07 million unfilled cybersecurity positions worldwide [1]. To meet the increasing demand for cybersecurity professionals, effective education and training are essential.

Modern educational environments for conducting cybersecurity training allow to collect data concerning trainees' activities throughout the training. These data have a high educational value. Their analysis can uncover insights about the trainees' learning processes and identify difficult sections of the training. Instructors can use this information to better design and structure cybersecurity training and improve assessment and guidance methods. However, manual transformation of these data into insights is not viable, since it is very time-consuming and ineffective [2, 3].

Data mining offers methods for the automatic extraction of useful information from the collected data [3]. One such method is called *pattern mining*. Pattern mining techniques, such as *association rule mining* and *sequential pattern mining*, can be used to discover interesting patterns in datasets [4].

The research aim of this thesis is to explore interesting patterns discovered in command logs from runs of a cybersecurity training. The patterns we want to discover and analyze are association rules and sequential patterns. Exploring association rules can provide insights into common mistakes and misconceptions, as well as identify concepts that are problematic for the trainees. Discovery and analysis of sequential patterns can help tutors observe conventional approaches to solving tasks of the training and reveal which sections of the training were the most challenging for the trainees.

This thesis is divided into six chapters. Chapter 2 presents an overview of the necessary theoretical background. Chapter 3 examines related research. Chapter 4 describes the analyzed data and the methods for data preprocessing, application of pattern mining algorithms, and postprocessing. Chapter 5 presents the extracted patterns and discusses their educational implications. Finally, Chapter 6 summarizes the thesis' findings and discusses the benefits and possibilities of further use and extension.

2 Theoretical Background

This chapter provides the theoretical background of this thesis. Section 2.1 introduces data mining and its subfield, pattern mining, to understand the context of this work. Subsequently, we focus on two pattern mining methods utilized in this thesis: association rule mining in Section 2.2 and sequential pattern mining in Section 2.3.

2.1 Data Mining

Data mining is a field of computer science that deals with extracting knowledge from data stored in databases. Its purpose is to understand the analyzed data better and to support decision-making based on the discovered results [5, 6]. There are many data mining methods, such as classification, clustering, outlier analysis, and pattern mining. These methods are used to discover new, interesting information in the given datasets [7, 5].

2.1.1 Pattern Mining

Pattern mining is an important data mining subfield. It deals with designing and implementing algorithms for automatic extraction of useful, previously hidden patterns in data. The main objective of pattern mining is to discover patterns that are easily interpretable by humans. Several pattern mining techniques exist, such as itemset mining, association rule mining, and sequential pattern mining [4, 5].

The interestingness of a pattern can be determined according to different criteria. For instance, some users may want to extract frequently occurring patterns (i.e., frequent patterns), while others may want to discover rarely occurring (i.e., rare) or long patterns [5, 4, 6].

The term *pattern mining* is more general than *frequent pattern mining*. Although in some cases, the two terms may be used interchangeably, *frequent pattern mining* only deals with discovering frequent patterns in datasets. In contrast, *pattern mining* also includes methods for discovering rare or negative[1] patterns [6].

1. A pattern is considered negative if it contains a negation of at least one item [5].

2.2 Association Rule Mining

Association rule mining belongs among the best-studied pattern mining techniques. Its goal is to discover association rules in a transaction database. Association rules are patterns with the form of an *if-then* statement. Association rule mining does not consider the ordering of items in transactions [5, 7].

2.2.1 Definition of Association Rule Mining

The following is a formal definition of an association rule [5, 6, 7].

- Let $I = \{I_1, I_2, \ldots, I_m\}$ be a set of literals called items.
- Let $D = \{T_1, T_2, \ldots, T_n\}$ be a set of transactions, also called *transaction database*, where each transaction T_i is a set of items such that $T_i \subseteq I$.

We say that a transaction T *contains* a set of items X if $X \subseteq T$. An association rule is a pattern of the form $X \rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$. We call X the *antecedent* and Y the *consequent*.

The *support* of a rule $X \rightarrow Y$ in the transaction database D is the proportion of transactions T_i that contain $X \cup Y$ among all transactions in D.[2] That is,

$$\text{support}(X \rightarrow Y) = \text{support}(X \cup Y) = \frac{|\{T_i : X \subseteq T_i \wedge Y \subseteq T_i\}|}{|D|}.$$

The *confidence* of a rule $X \rightarrow Y$ in the transaction database D is the proportion of transactions T_i that contain $X \cup Y$ among those transactions in D that contain X. That is,

$$\text{confidence}(X \rightarrow Y) = \frac{\text{support}(X \cup Y)}{\text{support}(X)} = \frac{|\{T_i : X \subseteq T_i \wedge Y \subseteq T_i\}|}{|\{T_i : X \subseteq T_i\}|}.$$

Given a transaction database D, the problem of mining association rules is to generate all association rules with support and confidence satisfying the user-defined minimum support (called *minsup*) and minimum confidence (called *minconf*). Association rules that satisfy both *minsup* and *minconf* thresholds are called *strong* rules [7, 6].

2. Support defined this way is also called *relative support*. Support defined as the total number of transactions T_i containing $X \cup Y$ is called *absolute support* [6].

2.2.2 Association Rule Mining Algorithms

There are many algorithms for association rule mining. The Apriori algorithm proposed in 1994 [8] is the most straightforward [7, p. 95]. Other notable algorithms include FP-Growth, MagnumOpus, and Closet. To use most of these algorithms, the user must first define the two previously mentioned thresholds *minsup* and *minconf* [7].

2.2.3 Measuring Interestingness of Association Rules

One of the challenges of applying association rule mining is to select "interesting" rules from all the extracted rules. Algorithms for mining association rules can extract a significant amount of rules depending on the thresholds for support and confidence. Filtering out the uninteresting ones can be a difficult task [7].

We can evaluate the interestingness of a rule using *objective interestingness measures*. To clarify, these measures evaluate the strength of correlation between the antecedent (X) and the consequent (Y) of the rule [6], not the overall usefulness of the rule to the user [7]. More than twenty measures, besides support and confidence, have been proposed [5]. These measures, however, may provide conflicting results about whether a given rule should be considered interesting [7].

Measures such as cosine, Jaccard, and all-confidence have the *null-invariant property*. This means that their calculation disregards the number of transactions that contain neither X nor Y (i.e., null-transactions). Consequently, the total number of transactions does not impact the resulting value of these measures [7, 6].

In [7, Chapter 17], it was observed that the three mentioned *null-invariant* measures rated *strong symmetric rules* as interesting. If both of the rules $X \rightarrow Y$ and $Y \rightarrow X$ exceed *minsup* and *minconf* thresholds, they are called *strong symmetric rules* [7, p. 246].

Measures such as lift, correlation, and chi-square are based on probability or statistics and do not have the null-invariant property. Because they are influenced by the total amount of transactions, they may produce conflicting results to the null-invariant measures depending on the characteristics of the transaction database [7, 6].

The recommended approach [7, p. 255] is first to measure the interestingness of the discovered rules with null-invariant measures.

Rules discarded by these measures should then be evaluated by a probability-based measure (e.g., lift). If a rule receives conflicting evaluations from the two types of measures, it is upon the user to decide whether to retain or discard the rule.

The following are definitions of three null-invariant interestingness measures: *cosine*, *Jaccard*, and *all-confidence*. Let us clarify that, similarly to the previous section, X and Y represent disjoint subsets of the set of items I.

Cosine [6, 7] is defined as

$$\text{cosine}(X \to Y) = \frac{\text{support}(X \cup Y)}{\sqrt{\text{support}(X) \times \text{support}(Y)}}.$$

Cosine is a number from 0 to 1, and an association rule is considered interesting if the number is above 0.65 [9].

Jaccard [7] is defined as

$$\text{Jaccard}(X \to Y) = \frac{\text{support}(X \cup Y)}{\text{support}(X) + \text{support}(Y) - \text{support}(X \cup Y)}.$$

Jaccard is a number from 0 to 1, and an association rule is deemed interesting if the number is above 0.5.

All-confidence [6, 7] is defined as

$$\text{all\_confidence}(X \to Y) = \frac{\text{support}(X \cup Y)}{\max(\text{support}(X),\ \text{support}(Y))},$$

where *max* denotes the maximum of the two numbers. All-confidence is a number in the range from 0 to 1, and an association rule is considered interesting if the number is above 0.5.

Lastly, we present the definition of a probability-based interestingness measure called *lift*. *Lift* [6, 7] is defined as

$$\text{lift}(X \to Y) = \frac{\text{support}(X \cup Y)}{\text{support}(X) \times \text{support}(Y)} = \frac{\text{confidence}(X \to Y)}{\text{support}(Y)}.$$

If lift is smaller than 1, X and Y are negatively correlated. Lift equal to 1 implies that there is no correlation between X and Y. A value greater than 1 suggests a positive correlation between X and Y.

2.3 Sequential Pattern Mining

Sequential pattern mining is a popular pattern mining technique used to discover interesting sequential patterns [5, 7]. These are frequently occurring subsequences in a given set of sequences.

2.3.1 Definition of Sequential Pattern Mining

We define a sequence as an ordered list of events, where each event is called an *item*[3]. Having two sequences of items, α, and β, we say that α is a *subsequence* of β if α is *contained* in β (denoted as $\alpha \sqsubseteq \beta$). For α to be *contained* in β, we must be able to form α from β by leaving out zero or more of β's items while preserving the relative order of the remaining items. For completeness, we say that β *contains* α [7, 5, 11, 12].

Let α be a sequence of items and S a set of sequences, also called *sequence database*. Support of α in S, denoted as support(α), is the number of sequences in S that *contain* α. Sequence α is a *sequential pattern* observed in S if support(α) $\geq$ *minsup* $\times |S|$, where *minsup* is the minimum support threshold specified by the user as percentage. In other words, α must occur in at least *minsup* percent of sequences in S to be recognized as a *sequential pattern* observed in S [7, p. 108].

Contrary to association rule mining, sequential pattern mining can be used to analyze data in which the ordering of items is relevant. Therefore, it is useful when recognizing sequential relationships between items could reveal potentially significant patterns [5].

2.3.2 Sequential Pattern Mining Algorithms

A multitude of algorithms exist for mining sequential patterns from a given sequence database, depending on the user-set *minsup* threshold. The most popular ones include GSP, Spade, PrefixSpan, Spam, Lapin, CM-Spam, and CM-Spade. It is important to notice that, given the same input, these algorithms all produce the same output. They differ only in the strategies and data structures utilized for discovering sequential patterns and, as a result, in their overall efficiency [5].

3. Elements of a sequence can also be defined as sets of items (i.e., itemsets), rather than just single items [10, 7]. For this thesis, we constrained sequences to contain single-item sets only because, in our data, two events cannot occur at the same time.

2.3.3 Sequential Pattern Mining Variations

This section explores some limitations of sequential pattern mining, as well as its variations suggested to address these limitations [5].

One major limitation of sequential pattern mining is discovering too many patterns, depending on the characteristics of the database and the user-set *minsup* threshold. This limitation hinders the performance of the pattern mining algorithms and users' ability to analyze the discovered patterns. The solution introduced to mitigate this limitation is *concise representations* of sequential patterns. A concise representation serves as a meaningful summary of all discovered sequential patterns. The following are the descriptions of three main concise representations of sequential patterns [5].

Maximal sequential patterns [5, 13] are sequential patterns that are not contained in other sequential patterns. That is, a sequential pattern s_a is *maximal* if no other sequential pattern s_b exists, such that $s_a \sqsubseteq s_b$. Algorithms for mining maximal patterns include VMSP and MaxSP.

Closed sequential patterns [5, 11, 12] are sequential patterns that are not contained in other sequential patterns with equal support. Formally, given a sequence database S, sequential pattern s_a is *closed* if no other sequential pattern s_b exists, such that $s_a \sqsubseteq s_b$ and their support in S is equal. Algorithms for mining closed sequential patterns include CloFast, Clasp, and CM-Clasp.

Generator sequential patterns [5, 14] (also called *sequential generators*) are sequential patterns that do not contain other sequential patterns with equal support. Formally, given a sequence database S, we say that a sequential pattern s_a is a *generator* if no other sequential pattern s_b exists, such that $s_b \sqsubseteq s_a$ and their support in S is equal. Algorithms for mining sequential generators include VGEN, FEAT, and FSGP.

Another limitation of sequential pattern mining is the necessity to set the *minsup* threshold accurately. This may prove difficult, especially if the user has no background knowledge about the database. If the *minsup* threshold is set incorrectly, it may negatively affect the number of patterns discovered[4] and the algorithms' performance. *Top-k sequential pattern mining* addresses this limitation. It lets the user set the number of k frequent sequential patterns to be discovered [5].

4. If *minsup* is too low, the number of extracted patterns may be too high. Setting the threshold too high, however, may result in discovering too few patterns [5].

3 State of the Art

This chapter provides an overview of related work. Section 3.1 introduces the field of educational data mining. Section 3.2 presents works that applied pattern mining to educational data. Section 3.3 examines studies that analyzed data from cybersecurity training.

3.1 Educational Data Mining

This section examines educational data mining (EDM), a growing interdisciplinary research area. Its objective is to create novel methods and algorithms to explore data from educational settings. The goal is to discover previously hidden information, which can be useful to understand student behavior and learning environments [15, 16].

The EDM process transforms data collected from educational environments into useful and easily comprehensible information used to improve the student learning experience. This process is similar to applications of data mining in other fields since it follows the same steps: preprocessing, application of data mining algorithms, and post-processing [17].

Several surveys and reviews of EDM have been published. Peña-Ayala [16] surveyed EDM works and described their approach by several characteristics. These characteristics include the model built from the collected data, tasks used to implement these models, and algorithms used.

Romero and Ventura [2] reviewed the state of the art of EDM and learning analytics (LA), which is a related research area. The survey listed important books, papers, journals, and conferences of these two affiliated research areas. It also described popular EDM and LA methods and their key applications, as well as trends and future objectives.

3.2 Pattern Mining in Educational Data

In this section, we present multiple studies that utilized association rule mining or sequential pattern mining. These two techniques are

applied to educational data mainly to identify learner difficulties, correlations between learning behaviors and performance, and teaching strategies that lead to better learning [2, 18].

García et al. [7, Chapter 7] applied association rule mining to data collected from a learning management system, which concerned students' usage of the system. They demonstrated that association rule mining can extract valuable insights from these data, such as relationships between students' activities and final marks. They suggested that instructors can use this information to adjust the course or identify struggling students.

Kobayashi [19] used association rule mining to uncover the types of errors that frequently co-occurred at various proficiency levels of spoken English students. Their goal was to discover which types of mistakes can distinguish lower-level from upper-level students using multiple data mining methods.

Malekian et al. [20] applied sequential pattern mining to data collected from an online learning environment. These data consisted of sequences of students' learning actions and assessment task submissions. The set of sequences was split into two categories, depending on the outcome of the sequence's final submission. They wanted to discover the preparation behavior patterns that lead to successful and unsuccessful assessment outcomes. Their results have shown that failed sequences contained mainly repeated assessment submissions and discussion forum views, while passed sequences consisted of multiple views and reviews of lecture materials. They suggested that this information can be used to modify the learning environment to discourage behavior that leads to unsuccessful learning outcomes.

3.3 Analysis of Data from Cybersecurity Training

This section summarizes several studies that analyzed data collected from cybersecurity training.

Weiss et al. [21] sought to provide better feedback to students by incorporating information about the exact steps students made, rather than just a numerical score. To achieve this, they developed a cloud-based framework for cybersecurity exercises named EDURange. This framework enabled them to extract command histories from students

who participated in the cybersecurity exercises. These data were used to model students' paths through the exercise, capturing the process of finding the solution. Compared to written exams, which can be very time-consuming to grade, this approach has immense potential for automatization.

Over the following years [22], EDURange was extended with several cybersecurity training scenarios. A utility for automatic generation of graphs from command histories was also implemented. Conclusively, this work demonstrated that command logs of students contain valuable data that can be used for both improving the assessment process as well as providing guidance to the students.

Abbott et al. [23] described an infrastructure for the automated collection of logs from cybersecurity training and parsing these logs into meaningful blocks of activity. The resulting dataset was statistically analyzed. This research was motivated by the notion that measuring performance based on achieved scores within the cybersecurity training is insufficient. Instead, valuable insights can be gained by analyzing the trainees' work processes and the tools they utilize.

McClain et al. [24] further analyzed the dataset described in [23] and combined it with questionnaires measuring the training participants' previous experience in cybersecurity. Their motivation was to uncover the differences in behavior and performance between cybersecurity beginners and professionals. They discovered that more experienced participants used specialized cybersecurity tools in combination with general-purpose tools, while the less experienced participants focused more heavily on specialized tools.

Mirkovic et al. [25] created a system named ACSLE for the collection and analysis of terminal input and output from participants in hands-on exercises. The system compares the collected data with pre-defined exercise milestones and produces statistics about the participants' progress. The system was evaluated on four hands-on cybersecurity exercises. It was shown that ACSLE's output helped identify difficult sections of the exercises and participants in need of assistance. This information is useful for instructors who can use it to provide timely interventions.

4 Methods

This chapter presents an overview of the methods used for our research. Section 4.1 introduces the infrastructure for organizing cybersecurity training runs. It also describes these training runs and data collected from them. Section 4.2 explains data preprocessing used to transform data into a format suitable for the utilized pattern mining algorithms. Section 4.3 presents the libraries used to extract patterns from data and explains postprocessing of the discovered results.

4.1 Research Environment

This section describes the infrastructure for cybersecurity training and the training's format. The data collection process and format of the collected data is also introduced.

4.1.1 Technical Infrastructure

KYPO cyber range [26, 27, 28] is a cloud-based platform capable of emulating complex networks through virtualization. It serves as a controlled environment for executing cyber attacks and practicing cybersecurity skills through training runs. KYPO cyber range is developed at Masaryk University.

4.1.2 Cybersecurity Training Format

The data analyzed in this thesis were collected from trainees who participated in KYPO training runs. In these runs, the trainees take on the role of an attacker in a network with one or more vulnerable hosts. Each trainee controls a virtual machine to attack the vulnerable hosts. The trainee's virtual machine comes pre-installed with Kali Linux: a penetration testing distribution [29] that provides all programs necessary to complete the training run.

To finish the training run, the trainee must complete several levels. Each level assigns a specific task, at the end of which the trainee obtains a string called a *flag*. Recovery and submission of this string is

necessary to advance into the next level. The trainees may also display several hints for each level, as well as a recommended solution.

4.1.3 Data Collection

KYPO cyber range allows trainees to enhance their cybersecurity skills in a variety of training assignments. The assignments that use command-line tools enable us to collect a *command log* [30] (i.e., *command history*) of each trainee.

Usually, about 10–20 trainees attend a single KYPO training instance, which takes up to two hours to complete. During this time, each trainee inputs 100–150 commands on average.

The data used in this thesis were collected from the first two levels of the training "House of Cards". A total of 22 command histories were collected, containing 1261 log records in total. They were collected from high school students and undergraduates attending the *KYPO Summer School* in August 2019 and students of Bachelor's and Master's degree programs attending the *PV276 Seminar on Cyber Attacks* course at Faculty of Informatics of the Masaryk University in September 2019.

The collected data are anonymized and do not include any personal information that could be used to discover the trainee's identity or track the trainee's progress throughout multiple training runs.

4.1.4 Data Format

The command history of each trainee is captured in a single JSON or JSONL file and consists of several log records. Each log record represents a single command executed by the trainee. An example of a log record collected from a KYPO training run is shown in Figure 4.1.

Each log record has a fixed number of attributes. For our purposes, the most significant are: `timegenerated`, representing the time of the command's execution, `message`, which represents the input command, and `app_name`, representing the application used to execute the command. The `app_name` attribute's value can either be "command", meaning that the command was executed in the Linux terminal, or "msf4-history", if the command was executed in the Metasploit [31] tool. The used application affects the format of `message` attribute's value. The `message` attribute generated by executing a command via Metasploit

```
{
    "timegenerated" : "2019-08-14T15:52:13.786137+02:00",
    "message"       : "set RPORT 23",
    "fromhost_ip"   : "127.0.0.1",
    "hostname"      : "attacker",
    "app_name"      : "msf4-history",
    "programname"   : "msf4-history",
    "procid"        : "-",
    "facility"      : "17",
    "severity"      : "6",
    "timereported"  : "2019-08-14T15:52:13.786137+02:00",
    "file"          : "\/root\/.msf4\/history"
}
```


Figure 4.1: Log record collected from a KYPO training run.

is shown in Figure 4.1. Example of a `message` attribute generated by executing a command via Linux terminal is shown in Figure 4.2.


```
{
    "message" : "src=\"172.16.1.120\"
                 uname=\"root\"
                 uid=\"0\"
                 wd=\"\/root\"
                 cmd=\"man nmap\"
                 v=\"1\""
}
```


Figure 4.2: Example of a `message` attribute generated by executing a command via Linux terminal.

The `message` attribute's value from a terminal command consists of several entries. The key information is stored in the entry `cmd`. It represents the full command executed, including the tool and arguments used. Other entries include `src`, which denotes the IP address of the machine on which the command was executed, and `uname`, the user account. Both `src` and `uname` usually remain the same throughout the training run. The `uname` is typically set to "root", as Kali Linux adhered to the "default root user policy" at the time of data collection.

4.2 Data Preprocessing

Before applying pattern mining algorithms to our data, we needed to preprocess the data to a suitable format. We implemented Python scripts to transform the collected command histories into an internal representation. This representation was then used to create several transaction databases and sequence databases described below.

4.2.1 Transaction Databases

We created two transaction databases that were used as input for association rule mining. The first database, called the *command transaction database*, represents input commands as separate transactions. Each transaction contains four items, representing the attributes of the command. These attributes are listed in Table 4.1.

Name	Description
TOOL	The tool used as part of the input command.
ARGS	The arguments used as part of the input command.
APP	The application used to execute the command.
DELAY	The time diffrence between this command and the following command, discretized to one of the values from the set {low, medium, high, undefined}.

Table 4.1: Attributes of input commands in the transactions of the *command transaction database*.

We employed two specialized preprocessing tasks to create this database. The first is to identify and resolve cases where a single trainee repeatedly inputs an identical command multiple times in succession. These cases need to be reduced to a single transaction[1] in the database. Otherwise, we might have discovered misleading association rules.

The second preprocessing task, inspired by [7, Chapter 7], is the *discretization* of the `delay` attribute. *Discretization* is the process of dividing continuous data into categorical classes to increase the interpretability and comprehensibility of the data [7, p. 102]. Several

1. In these cases, we used the timestamp of the first command in the succession to compute the value for the `delay` attribute.

discretization methods exist, such as the equal-width method, the equal-frequency method, and the manual method, where the user sets the cut-off points for individual categories [7, p. 102]. We used a self-implemented function to compute the cut-off points for the `delay` attribute based on the average[2] delay time.

Before discretization, the delay value in seconds was computed for each command[3]. Our data contained a small number of disproportionately high delay values due to some trainees solving the training tasks over multiple days. To resolve the problem of outlying delays, a maximum of 20 minutes (1200 seconds) was set. This approach was inspired by [32]. Delays exceeding the maximum were discretized to "undefined". The intervals for "low", "medium", and "high" categories were computed based on the mean delay from all delays not exceeding the pre-defined maximum.

The second database, called the *tool transaction database*, contains transactions consisting of only two attributes: `tool` and `delay`. This database was created by merging the consecutive uses of the same tool into a single transaction. The `delay` attribute represents the time gap between the first use of a tool and the next use of a different tool. Delay values were discretized as in the first transaction database. The motivation behind creating this database was to uncover the level of difficulty associated with different tools. Should a tool be associated with long delay times, it may indicate that the trainees were unfamiliar with this tool and had difficulties using it.

Table 4.2 shows the distribution of delays of each transaction database across the four mentioned categories. Since the dataset was skewed toward lower delay times, most `delay` attributes in both transaction databases were discretized to the "low" value.

Database	Low	Medium	High	Undefined
command transaction database	739	211	199	44
tool transaction database	525	160	159	43

Table 4.2: The distribution of delays across the four categories.

2. We decided to use the mean delay value instead of median, because the cut-off points computed based on the median value were too low.
3. The last command in each command history was assigned the value infinity.

4.2.2 Sequence Databases

Three sequence databases were created as input for sequential pattern mining algorithms. All these databases consisted of an equal number of 22 sequences (matching the number of command histories), differing only in the items contained within these sequences.

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

The second database, referred to as *tool sequence database*, contains sequences of tools used by the training participants. Each item represents a tool used within a command by the trainee.

Let us clarify that commands/tools both from terminal and Metasploit are included together in the sequences of these two sequence databases. This allows us to discover longer patterns, which more accurately reflect the trainees' progress throughout the training.

The third database, *application sequence database*, stores sequences of applications utilized by the trainees to execute commands. This database contains only two unique items: *terminal* and *metasploit*. Table 4.3 shows the number of transactions/sequences and unique items contained in each of our databases.

Database	Transactions/sequences	Unique items
command transaction database	1193	419
tool transaction database	887	96
command sequence database	22	415
tool sequence database	22	92
application sequence database	22	2

Table 4.3: The number of transactions/sequences and unique items contained in each created database.

4.3 Data Analysis

This section presents data mining libraries used to extract patterns from databases introduced in Section 4.2. Moreover, it describes how the discovered patterns are visualized.

4.3.1 Application of Pattern Mining Algorithms

We used the Apyori [33] Python library and the SPMF [34] library to apply pattern mining algorithms on the created databases.

We used the Apyori library for association rule mining. Apyori is a compact Python module that implements the Apriori algorithm. We chose this library because it seamlessly integrates with our self-implemented null-invariant measures for association rules (see Section 2.2.3). The SPMF library does not implement these measures. Other advantages of the Apyori library include its ease of use and independence on other libraries, which enables its portability [33].

For sequential pattern mining, we used SPMF: an open-source data mining library specialized for pattern mining. It provides optimized and documented implementations of more than 190 data mining algorithms [35] often used as benchmarks in research papers [34].

We selected *Clofast*, an efficient algorithm [12] for mining *closed sequential patterns* (see Section 2.3.3). Mining closed sequential patterns has several benefits. Firstly, the set of extracted patterns is smaller, while serving as a lossless[4] representation of all sequential patterns [5]. Thus, the discovered patterns are easier to visualize and analyze. Moreover, closed sequential patterns "represent the largest subsequences common to sets of sequences" [5]. This fact is important because we are interested in extracting longer patterns, which more closely represent the trainees' progress. Many previous studies support mining closed sequential patterns instead of all sequential patterns [12].

The *minsup* threshold was manually tuned for each database. Since there is no simple method to determine the best threshold [36], this was done by trial and error. The threshold was initially set to higher values and then gradually lowered until we reached a number of patterns manageable for interpretation and comprehensible visualization.

In association rule mining, the *minconf* threshold, compared to the *minsup* threshold, is generally easier to set. This is because the database's properties that are often unknown to the user more heavily influence *minsup* compared to *minconf* [37].

In our research, we were interested in rules with higher confidence and therefore used higher *minconf* thresholds. In contrast, the *minsup*

4. The set of all sequential patterns and their support values can be retrieved from the set of closed sequential patterns without scanning the database [5].

thresholds for association rule mining needed to be much lower to extract a sufficient amount of rules. This was most likely because our transaction databases contained relatively many unique items when considering the total amount of transactions in the databases.

4.3.2 Data Postprocessing

In the data postprocessing phase, we visualize larger sets of patterns and interpret and evaluate the results. The educational insights gained after this phase can then be used by tutors to improve the training.

The discovered patterns were visualized using the Plotly [38] Python library. The association rules were visualized through a *bubble chart*, as inspired by [39]. In our bubble chart, each extracted rule is represented by a bubble. The size of the bubble represents the support of the rule, and the marker's color represents the rule's confidence.

The sequential patterns were visualized using *Sankey diagrams* inspired by [40]. The items of the discovered sequential patterns are represented as nodes. Edges between the nodes represent subsequences of the sequential patterns. Common subsequences are not combined into a single edge, but are shown as individual edges. As a result, the total thickness of multiple edges visually distinguishes the prominent subsequences. The thickness of individual edges varies depending on the support of the pattern in which the subsequence occurs. Sequential patterns consisting of only one item are excluded from the diagrams.

Figure 4.3 summarizes the inputs and outputs of the three phases of transforming the command logs into educational insights.

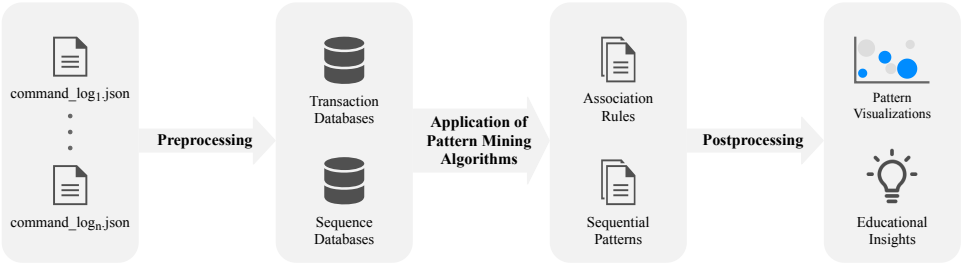

Figure 4.3: Diagram representing inputs and outputs of *preprocessing*, *application of pattern mining algorithms*, and *postprocessing* phases.

5 Results and Discussion

This chapter lists and interprets patterns mined from the databases described in Section 4.2. Section 5.1 presents and discusses the patterns discovered in our *transaction databases*. Section 5.2 presents and discusses the patterns extracted from our *sequence databases*. Larger sets of discovered patterns are visualized. Additionally, interesting subsets of the discovered patterns are tabulated. Complete sets of discovered patterns can be found in the files described in Appendix A.

5.1 Transaction Databases

This section presents the association rules discovered in transaction databases introduced in Section 4.2.1. Association rules presented in all tables have the *antecedent* and *consequent* separated by "$\rightarrow$". All decimal numbers in the tables are rounded to two decimal places. The tables contain the *absolute support* values of rules (see Section 2.2.1).

5.1.1 Command Transaction Database

We now present the association rules discovered in the *command transaction database*. The *minsup* threshold was set to 0.04 and *minconf* was set to 0.6. In total, 43 association rules were generated for these thresholds. The rules describe the relationships between the attributes of commands. Figure 5.1 visualizes these rules.

Eight of the discovered rules were labeled as interesting by the null-invariant measures introduced in Section 2.2.3. These eight association rules are presented in Table 5.1.

The first two rules in Table 5.1 concern commands executed in Metasploit. These two rules have the highest support from all discovered rules (see Figure 5.1). The first rule tells us that 74% of commands executed in Metasploit correlate with low delay times. The second rule indicates that 71% of commands associated with low delays were executed in Metasploit.

The remaining rules (3–8) in Table 5.1 concern commands containing the tool `show`. This tool is used in Metasploit, mainly to display available modules and their options [31].

Figure 5.1: Visualization of all association rules mined from *command transaction database* for *minsup*=0.04 and *minconf* =0.6. The size of each marker represents the rule's support, and the marker's color represents the confidence of the rule.

#	Rule	Sup	Con	Cos	Jac	A-c
1	APP=metasploit → DELAY=low	527	0.74	0.72	0.57	0.71
2	DELAY=low → APP=metasploit	527	0.71	0.72	0.57	0.71
3	ARGS=['options'] → TOOL=show	51	1.00	0.86	0.74	0.74
4	TOOL=show → ARGS=['options']	51	0.74	0.86	0.74	0.74
5	ARGS=['options'] → TOOL=show APP=metasploit	51	1.00	0.86	0.74	0.74
6	TOOL=show → APP=metasploit ARGS=['options']	51	0.74	0.86	0.74	0.74
7	APP=metasploit ARGS=['options'] → TOOL=show	51	1.00	0.86	0.74	0.74
8	TOOL=show APP=metasploit → ARGS=['options']	51	0.74	0.86	0.74	0.74

Table 5.1: Association rules mined from *command transaction database* rated as interesting by null-invariant-based measures (*minsup*=0.04, *minconf* =0.6). *Sup*, *Con*, *Cos*, *Jac*, and *A-c* are abbreviations of interestingness measures described in Section 2.2.

Rule 3 implies that the argument `options` was used only in combination with the tool `show`, while rule 4 tells us that in 74% of uses, `show` was executed with the argument `options`. The command `show options` displays settings of the selected module [31].

Rules 5–8 are similar to rules 3 and 4. They differ only in the inclusion of "APP=metasploit" in either antecedent or consequent of the rule. However, since the support and confidence of the rule remain unchanged, it is apparent that commands containing the tool `show` were executed exclusively via Metasploit.

Table 5.2 presents ten rules rated as uninteresting by cosine, Jaccard, and all-confidence, which have the highest support values and lift value greater than one. Interestingly, of all 43 discovered association rules, 42 had the lift value greater than one.

#	Rule	Sup	Con	Lift
1	ARGS=[] → APP=metasploit	263	0.74	1.23
2	TOOL=set → APP=metasploit	218	0.98	1.63
3	TOOL=set → DELAY=low	202	0.91	1.46
4	APP=metasploit TOOL=set → DELAY=low	198	0.91	1.47
5	DELAY=low TOOL=set → APP=metasploit	198	0.98	1.63
6	TOOL=set → DELAY=low APP=metasploit	198	0.89	2.01
7	DELAY=low ARGS=[] → APP=metasploit	159	0.77	1.28
8	DELAY=high → APP=terminal	127	0.64	1.60
9	TOOL=ssh → APP=terminal	112	1.00	2.51
10	TOOL=nmap → APP=terminal	99	1.00	2.51

Table 5.2: Ten association rules (not included in Table 5.1) mined from *command transaction database* with the highest support values and lift value greater than one (*minsup*=0.04, *minconf*=0.6).

We now describe Table 5.2. Rule 1 shows that 74% of commands without arguments were executed via Metasploit. Rules 2–6 concern the tool `set`, which configures the options of the selected module [31] in Metasploit. These rules show that commands containing this tool were largely executed in Metasploit with low delay times. Rule 7 tells us that 77% of commands without arguments and with low delay times were executed in Metasploit. Rule 8 indicates that 64% of high-delay commands are executed via terminal. Lastly, rules 9 and 10 show

that commands containing the tools `ssh` and `nmap` were executed solely via Linux terminal.

5.1.2 Tool Transaction Database

For mining in the *tool transaction database*, we set the *minsup* and *minconf* thresholds to 0.01 and 0.5 respectively. Ten association rules were discovered, which describe the correlations between tools and delays they introduce. However, none of the discovered rules was labeled as interesting by the null-invariant measures. Reasons for this are discussed in Section 5.1.4. Table 5.3 shows the rules mined from *tool transaction database* with lift value higher than one.

#	Rule	Sup	Con	Lift
1	TOOL=nmap → DELAY=high	24	0.51	2.85
2	TOOL=exit → DELAY=low	17	0.63	1.06
3	TOOL=exploit → DELAY=low	26	0.60	1.02
4	TOOL=options → DELAY=low	34	0.85	1.44
5	TOOL=set → DELAY=low	127	0.87	1.47
6	TOOL=show → DELAY=low	46	0.77	1.30
7	TOOL=use → DELAY=low	62	0.89	1.50

Table 5.3: Association rules mined from *tool transaction database* with lift greater than one (*minsup*=0.01, *minconf* =0.5).

The first rule, which has the highest lift value among these seven rules, tells us that the `nmap` tool is associated with high delays in 51% of `nmap` uses. The remaining rules (2–7) concern tools correlated with low delay times. These tools are typically used in Metasploit commands.

5.1.3 Discussion

The following is an interpretation of the discovered association rules and evaluation of insights that emerged from the results.

A considerable amount of rules indicate a correlation between commands executed in Metasploit and low delays. This fact could imply that trainees were more inclined to experiment with these commands.

Another possible interpretation is that Metasploit commands were easier to understand and use for the trainees.

In contrast, the correlation between high delays and terminal-executed commands may suggest that using such commands is more difficult. It is important, however, to consider other factors affecting these results. For instance, the fact that commands without arguments were mostly executed in Metasploit could be one of the reasons for the association between Metasploit commands and low delay times. Naturally, commands executed without arguments should require less time spent on reading their documentation.

Although a strong association between a delay class and a particular command or tool may indicate its difficulty of use, other possible causes for such correlation should be considered. For instance, the correlation between the network scanning tool, `nmap`, and high delays can also stem from the scan's duration, which depends on the used arguments, as observed in [21].

The high support of some high-confidence rules can reveal the overuse of certain tools. For example, while rules 2, 9, and 10 from Table 5.2 confirm that generally, trainees used the correct application (terminal or Metasploit) to employ the given tools, the high support of these rules suggests excessive use of the tools. When compared to the recommended solution, commands utilizing `set`, `run`, `nmap`, or `ssh` were, on average, executed more than four times the necessary amount. This indicates that some training participants had difficulties grasping these tools and using them correctly.

5.1.4 Lessons Learned

After the application of the three null-invariant measures, we can conclude that they do not perform particularly well on our association rules. The reason for this lies most likely in the type of association rules generated from our databases.

Merceron and Yacef [7, Chapter 17] stated that *cosine*, *Jaccard*, and *all-confidence* rate *strong symmetric rules* (see Section 2.2.3) as interesting. However, the majority of association rules mined from our databases was not strongly symmetric.

In contrast, the lift measure rated most of the discovered rules as interesting. Conclusively, while using the two types of measures had

limited effect on pruning our association rules, using them complementarily seems to be the correct approach, as it reduces the risk of discarding potentially important rules.

In the future, it might be worthwhile to consider other methods for decreasing the total amount of discovered rules, such as constraining the maximum amount of items in antecedent and consequent.

5.2 Sequence Databases

This section presents closed sequential patterns mined from sequence databases introduced in Section 4.2.2. These patterns reveal common subsequences discovered in sequences of the databases. Events (i.e., items) of patterns presented in tables are separated by "→".

5.2.1 Command Sequence Database

Closed sequential pattern mining with the *minsup* threshold set to 0.5 resulted in the discovery of 67 sequential patterns. Figure 5.2 visualizes these patterns (excluding the patterns composed of a single event).

Table 5.4 presents twelve patterns with the highest support mined from *command sequence database*. Most of these patterns contain either the command `set rhosts 172.18.1.5`, or `set rport 23`. The first two patterns show that almost every trainee executed these commands. These commands do not appear together in patterns with high support, probably because the trainees executed them in a different order.

Other prominent commands within these patterns are `run` and `msfconsole`. The former launches the selected exploit. The latter opens *MSFconsole*, a command-line interface of Metasploit [31]. The slight support decrease between patterns 1–2 and 3–4 most likely results from the command `exploit`, which is an alias of `run` [31]. The relatively small support of the command `msfconsole` could be caused by trainees launching the MSFconsole via the graphical user interface of Kali Linux. Pattern 8 shows that the command `run` was executed at least twice in 17 command histories. Pattern 10 shows that 16 of 22 trainees utilized the recommended exploit `libssh_auth_bypass`.

Table 5.5 presents the ten longest closed sequential patterns mined from *command sequence database*. These patterns contain the same com-

#	Pattern	Sup
1	set rhosts 172.18.1.5	22
2	set rport 23	21
3	set rhosts 172.18.1.5 → run	21
4	set rport 23 → run	20
5	set rhosts 172.18.1.5 → run → ssh alice@172.18.1.5	19
6	msfconsole	18
7	set rport 23 → run → ssh alice@172.18.1.5	18
8	run → run	17
9	msfconsole → set rhosts 172.18.1.5 → run	17
10	use . . . /libssh_auth_bypass → set rhosts 172.18.1.5	16
11	msfconsole → set rport 23 → run	16
12	msfconsole → set rhosts 172.18.1.5 → run → ssh alice@172.18.1.5	16

Table 5.4: Twelve closed sequential patterns with highest support mined from *command sequence database*. The command `use auxiliary/scanner/ssh/libssh_auth_bypass` was shortened to `use .../libssh_auth_bypass`.

mands as patterns in Table 5.4. Most of them begin with the command `msfconsole`.

Two commands were used repeatedly in multiple patterns in Table 5.5. These are the command `run` and `ssh alice@172.18.1.5`. Using `ssh alice@172.18.1.5` opened a new session at the beginning of the training's third level.

Patterns 6 and 7 in Table 5.5 show an interesting relationship between the commands `run` and `set rhosts 172.18.1.5`. According to these patterns, 11 trainees launched an exploit before setting the `rhosts` option. A similar relationship can be observed in Figure 5.2 between the commands `run` and `set rport 23`.

Furthermore, Figure 5.2 reveals other commands that appeared less frequently in the discovered patterns. These commands include `nmap -sV 172.18.1.5`, `search libssh`, and `ssh 172.18.1.5`.

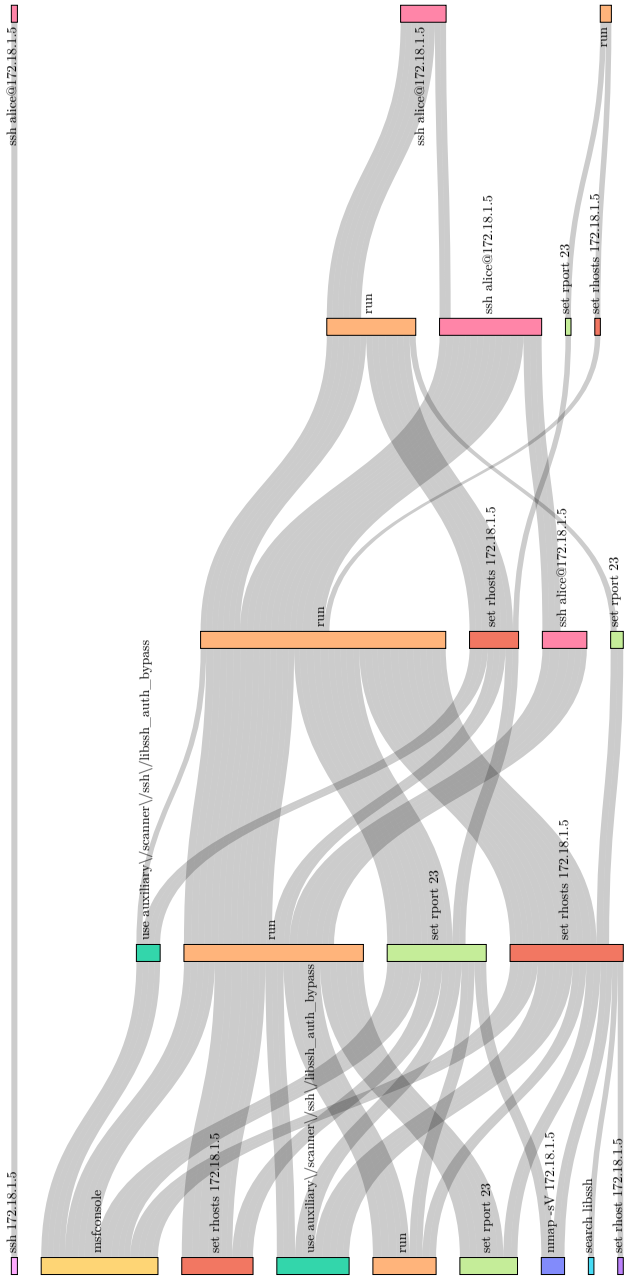

Figure 5.2: Visualization of the closed sequential patterns mined from *command sequence database* for *minsup*=0.5 (patterns consisting of single event are excluded).

#	Pattern	Sup
1	msfconsole → set rport 23 → set rhosts 172.18.1.5 → run → ssh alice@172.18.1.5	12
2	msfconsole → use . . . /libssh_auth_bypass → set rhosts 172.18.1.5 → run → ssh alice@172.18.1.5	12
3	set rhosts 172.18.1.5 → run → ssh alice@172.18.1.5 → ssh alice@172.18.1.5 → ssh alice@172.18.1.5	11
4	set rhosts 172.18.1.5 → run → run → run → ssh alice@172.18.1.5	11
5	msfconsole → set rhosts 172.18.1.5 → run → ssh alice@172.18.1.5 → ssh alice@172.18.1.5	11
6	msfconsole → set rport 23 → run → set rhosts 172.18.1.5 → run	11
7	msfconsole → run → set rhosts 172.18.1.5 → run → ssh alice@172.18.1.5	11
8	msfconsole → run → run → run → ssh alice@172.18.1.5	11
9	msfconsole → use . . . /libssh_auth_bypass → set rhosts 172.18.1.5 → set rport 23 → run	11
10	msfconsole → use . . . /libssh_auth_bypass → run → run → ssh alice@172.18.1.5	11

Table 5.5: Ten longest closed sequential patterns mined from *command sequence database* with *minsup*=0.5. The command `use auxiliary/scanner/ssh/libssh_auth_bypass` was shortened to `use .../libssh_auth_bypass`.

5.2.2 Tool Sequence Database

A total of 147 *closed sequential patterns* were discovered in *tool sequence database* with the *minsup* threshold set to 0.6. Figure 5.3 visualizes these patterns.

Ten of the discovered patterns with the highest support are presented in Table 5.6. Interestingly, all except the second pattern contain the tool `nmap` as the first event. Pattern 1 confirms that all trainees utilized this tool. Furthermore, Figure 5.3 also shows that most discovered patterns begin with the `nmap` tool. This contrasts with the patterns mined from *command sequence database*, where the commands containing `nmap` represent only a small fraction (see Figure 5.2).

Pattern 2 shows that the tool `ssh` was used at least twice by 21 trainees. Patterns 3–10 show mostly similar sequential relationships: the execution of `use` followed by `set` and `run`, and the use of `nmap` at the beginning and `ssh` at the end of patterns.

#	Pattern	Sup
1	nmap → set → set	22
2	ssh → ssh	21
3	nmap → use → set	21
4	nmap → set → set → ssh	21
5	nmap → set → set → run	21
6	nmap → set → set → set	20
7	nmap → use → set → ssh	20
8	nmap → use → set → run	20
9	nmap → set → set → ssh → ssh	20
10	nmap → set → set → run → ssh	20

Table 5.6: Ten closed sequential patterns with the highest support mined from *tool sequence database*.

Table 5.7 presents 16 longest closed sequential patterns discovered. These are similar to the patterns presented in Table 5.6, but generally contain more occurrences of the tools `set` and `run`. Interestingly, while most of these patterns contain multiple occurrences of `set` and `run`, they contain at most one occurrence of `use`. Figure 5.3 shows that `use` appears as the second event in most patterns containing this tool.

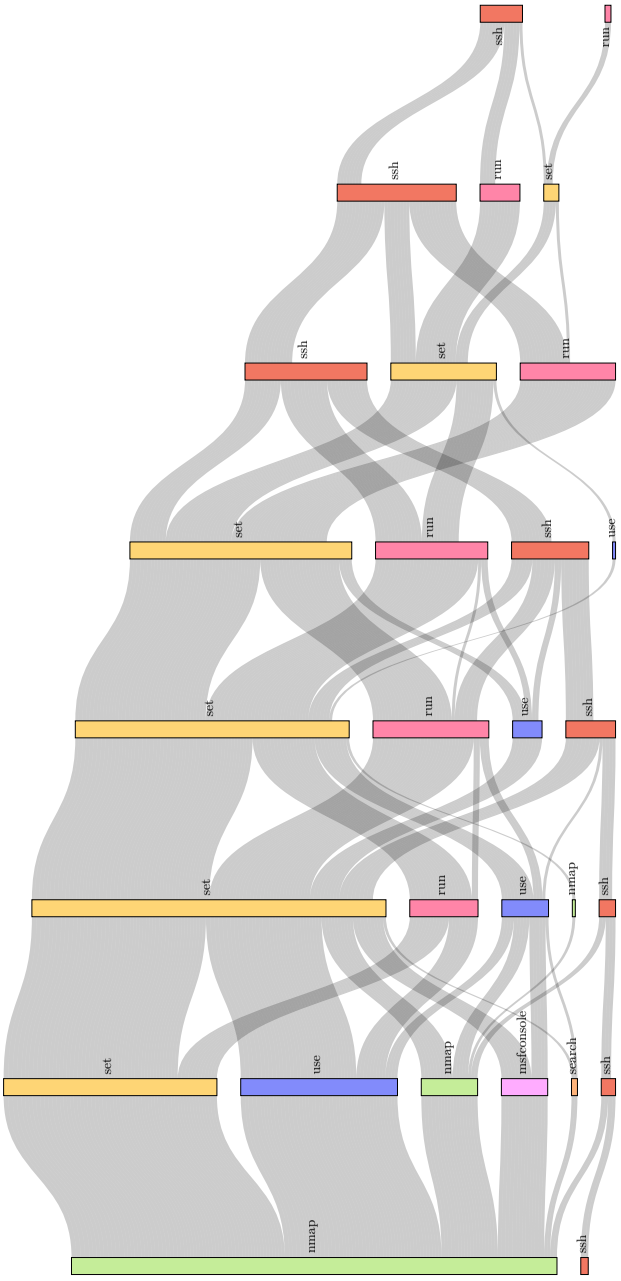

Figure 5.3: Visualization of closed sequential patterns mined from *tool sequence database* for *minsup*=0.6.

#	Pattern	Sup
1	nmap → msfconsole → set → set → set → run → ssh → ssh	15
2	nmap → set → set → set → set → set → run → ssh	15
3	nmap → set → run → run → set → run → ssh → ssh	14
4	nmap → set → run → set → run → set → run → ssh	14
5	nmap → set → set → run → set → run → ssh → ssh	14
6	nmap → set → set → run → set → set → run → ssh	14
7	nmap → set → set → run → set → set → set → run	14
8	nmap → set → set → run → set → set → set → ssh	14
9	nmap → set → set → set → set → run → set → run	14
10	nmap → set → set → set → set → run → ssh → ssh	14
11	nmap → set → set → set → set → set → ssh → ssh	14
12	nmap → use → set → run → run → set → run → ssh	14
13	nmap → use → set → run → set → run → ssh → ssh	14
14	nmap → use → set → set → run → set → run → ssh	14
15	nmap → use → set → set → run → set → ssh → ssh	14
16	nmap → use → set → set → set → run → ssh → ssh	14

Table 5.7: Sixteen longest closed sequential patterns mined from *tool sequence database* with *minsup*=0.6.

5.2.3 Application Sequence Database

Closed sequential pattern mining with the *minsup* threshold set to 0.7 extracted 46 patterns visualized in Figure 5.4.

From the discovered patterns, Table 5.8 shows nine with the highest support. Pattern 1 confirms that all trainees utilized both applications during the training. What is more, it shows that every trainee executed at least six commands through a Linux terminal, and seven commands through Metasploit. Pattern 5 shows that 20 training participants executed a minimum of 13 commands via terminal. A common trend among all but the fifth pattern is that they show two series of terminal commands interrupted by a series of Metasploit commands.

Table 5.9 presents eight longest of the 46 discovered closed sequential patterns. These patterns show a similar trend to those in Table 5.8. Interestingly, when we compare the two subsets of patterns, we discover that most patterns in Table 5.9 contain more terminal commands while having a similar amount of Metasploit commands.

5.2.4 Discussion

The following is an interpretation of the extracted closed sequential patterns and evaluation of insights provided by these results. Various patterns may provide different insights. Repetition of a command or a tool may suggest difficulties with its usage. Patterns with high support reveal commonly used commands and tools, while longer patterns may expose common strategies among training participants.

The patterns mined from *command sequence database* introduce a number of potentially significant insights. The majority of these commands show the selection, configuration, and launching of an exploit. The only exploit present in these patterns is the `libssh_auth_bypass`, which suggests that it was the most commonly used exploit. This strategy coincides with the recommended solution of the training.

The repeated occurrence of the command `run` in patterns mined from *command sequence database* and *tool sequence database* suggests that trainees might have had difficulties with launching exploits. To clarify, it was necessary to execute only one exploit.

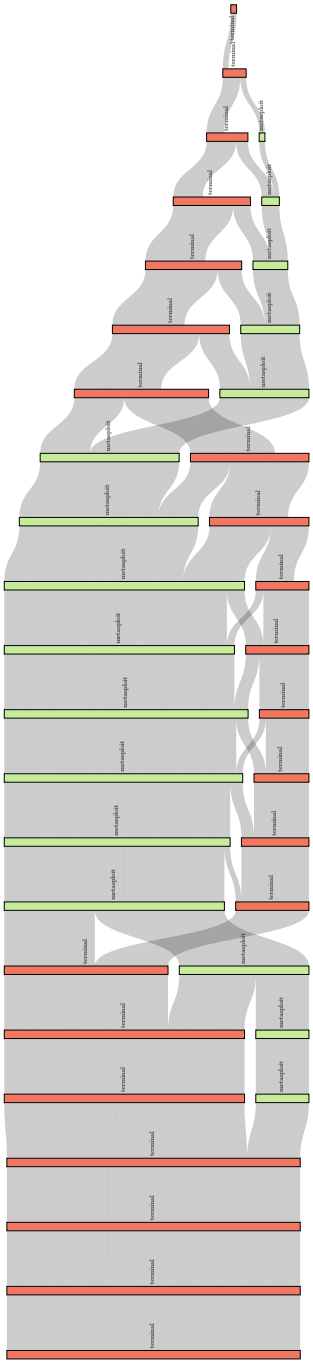

Figure 5.4: Visualization of closed sequential patterns mined from *app sequence database* for *minsup*=0.7.

#	Pattern	Sup
1	T → T → T → T → M → M → M → M → M → M → M → T → T	22
2	T → T → T → T → M → M → M → M → M → M → M → M → T → T	21
3	T → T → T → T → T → T → M → M → M → M → M → M → M → T → T	21
4	T → T → T → T → T → T → T → M → M → M → M → M → M → T → T	21
5	T → T → T → T → T → T → T → T → T → T → T → T → T → T	20
6	T → T → T → T → T → T → T → M → M → M → M → M → M → M → M → T → T	20
7	T → T → T → T → T → T → T → M → M → M → M → M → M → M → M → T → T → T	20
8	T → T → T → T → T → T → T → T → M → M → M → M → M → M → M → T → T	20
9	T → T → T → T → T → T → T → T → M → M → M → M → M → M → T → T → T	20

Table 5.8: Nine closed sequential patterns with highest support mined from *application sequence database*. "Terminal" and "Metasploit" were shortened to "T" and "M" respectively.

#	Pattern	Sup
1	T → T → T → T → T → T → T → T → T → T → T → T → M → M → M → M → M → M → M → M → T → T → T	17
2	T → T → T → T → T → T → T → T → T → T → T → T → T → T → M → M → M → M → M → M → M → M → T	17
3	T → T → T → T → T → T → T → T → T → T → T → T → M → M → M → M → M → M → M → M → T → T → T → T → T	17
4	T → T → T → T → T → T → T → T → M → M → M → M → M → M → M → M → M → T → T → T → T → T → T → T	16
5	T → T → T → T → T → T → T → T → M → M → M → M → M → M → M → M → M → T → T → T → T → T → T	16
6	T → T → T → T → T → T → T → T → T → T → T → T → T → T → M → M → M → M → M → M → M → M → M → T	16
7	T → T → T → T → T → T → T → T → T → T → T → T → T → T → M → M → M → M → M → M → M → M → T → T	16
8	T → T → T → T → T → T → T → T → T → T → T → M → M → M → M → M → M → M → M → T → T → T → T	16

Table 5.9: Eight longest closed sequential patterns mined from *application sequence database* with *minsup*=0.7. "Terminal" and "Metasploit" were shortened to "T" and "M" respectively.

The alternation between `set` and `run` in patterns from Table 5.7 implies difficulties with correctly setting all necessary options before launching an exploit. Patterns in Table 5.5 and Figure 5.2 that show the execution of `run` before setting the `rhosts` or `rport` option support this assumption.

The repetition of `ssh alice@172.18.1.5` in patterns 3 and 5 in Table 5.4 implies that a non-trivial amount of trainees struggled with using this command. What is more, the pattern `ssh 172.18.1.5` → `ssh alice@172.18.1.5` in Figure 5.2 supports the assumption that some trainees were unfamiliar with the `ssh` tool.

A vast majority of patterns discovered in *tool sequence database* begin with the tool `nmap`, which indicates that it was commonly used in the early stages of the training. What is more, the low occurrence of commands containing `nmap` in patterns from the *command sequence database* suggests that this tool was executed with a variety of different arguments.

The low occurrence of the tool `use` in commands from *tool sequence database* implies that training participants usually did not experiment with many different exploits.

The trend of two series of terminal commands interrupted by a series of Metasploit commands observed in patterns from *application sequence database* implies that, in general, players did not switch between these two applications very often, but instead used them in longer bursts.

The fact that longer patterns from *application sequence database* contain mainly more terminal commands suggests that the use of terminal commands might have been more difficult for the trainees than the use of Metasploit commands.

6 Conclusion

This thesis explored interesting patterns in command logs from cybersecurity training. We used two pattern mining techniques, *association rule mining* and *sequential pattern mining*, to extract patterns from command histories from 22 training runs.

The contribution of this thesis is the presentation of the discovered patterns, as well as their interpretation into insights about the trainees' learning processes. To the best of our knowledge, this was the first application of association rule mining and sequential pattern mining algorithms to command logs from cybersecurity training. Other instructors can also use these techniques to extract useful insights from their data and improve cybersecurity education.

The results confirm that pattern mining can reveal interesting educational insights in command histories from cybersecurity training. These insights include common mistakes, misconceptions, concepts problematic for the trainees, difficult sections of the training, and typical strategies and tools.

The discovered association rules suggest that commands executed through Metasploit correlate with low delay times, while commands executed through the Linux terminal correlate with long delay times. This could indicate difficulties with using terminal commands, however, other factors affecting the results need to be considered. Furthermore, high support of certain rules implies the overuse of commands containing the `set`, `run`, `nmap`, or `ssh` tool.

The extracted sequential patterns suggest that the trainees' strategy was often similar to the recommended solution. The trainees rarely experimented with different exploits. What is more, the discovered patterns also indicate difficulties with the execution of exploits and the use of commands containing the `ssh` tool.

The insights gained by analyzing command histories can improve cybersecurity training assignments in various ways. The instructors can use this information to identify difficult concepts or sections of the training. It can also be used to improve the existing hints or create new ones. Furthermore, insights concerning common strategies can reveal new solutions to the training exercise, which the instructors did not consider.

6.1 Future Work

In the future, we could apply the pattern mining techniques to a larger set of command histories from the same cybersecurity training. Doing so could reveal new patterns and insights or confirm our previous findings and hypotheses. These techniques may also be used to discover patterns in logs from other trainings to analyze and improve them.

Furthermore, the dataset used as input for the pattern mining algorithms could be expanded with training events, such as taking hints or finishing a training's level. Mining patterns in this expanded dataset might reveal additional interesting insights, such as the most frequently taken hints and common sequences of actions that preceded taking a hint or finishing a level.

Bibliography

[1] (ISC)^2. Cybersecurity Workforce Study. https://www.isc2.org/Research/2019-Cybersecurity-Workforce-Study/ [Online; accessed 2020-06-24], 2019.

[2] Cristobal Romero and Sebastian Ventura. Educational data mining and learning analytics: An updated survey. *WIREs Data Mining and Knowledge Discovery*, 10(3):e1355, 2020. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1355, doi:10.1002/widm.1355.

[3] Philippe Fournier-Viger. An Introduction to Data Mining. http://data-mining.philippe-fournier-viger.com/introduction-data-mining/ [Online; accessed 2020-06-24], Feb 2017.

[4] Philippe Fournier-Viger. An introduction to frequent pattern mining. http://data-mining.philippe-fournier-viger.com/introduction-frequent-pattern-mining/ [Online; accessed 2020-06-24], Oct 2013.

[5] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Rage Uday Kiran, Yun Sing Koh, and Rincy Thomas. A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1(1):54–77, 2017.

[6] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[7] Cristobal Romero, Sebastian Ventura, Mykola Pechenizkiy, and Ryan SJd Baker, editors. *Handbook of educational data mining*. CRC press, 2010.

[8] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th VLDB Conference*, pages 487–499, 1994.

[9] Agathe Merceron and Kalina Yacef. Interestingness Measures for Association Rules in Educational Data. *Educational Data Mining*, pages 57–66, 2008.

[10] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14. IEEE Comput. Soc. Press, 1995. doi:10.1109/icde.1995.380415.

[11] Philippe Fournier-Viger, Antonio Gomariz, Manuel Campos, and Rincy Thomas. Fast Vertical Mining of Sequential Patterns Using Co-occurrence Information. In *Advances in Knowledge Discovery and Data Mining*, pages 40–52. Springer International Publishing, 2014. doi:10.1007/978-3-319-06608-0_4.

[12] Fabio Fumarola, Pasqua Fabiana Lanotte, Michelangelo Ceci, and Donato Malerba. CloFAST: Closed Sequential Pattern Mining Using Sparse and Vertical Id-Lists. *Knowledge and Information Systems*, 48(2):429–463, August 2016. doi:10.1007/s10115-015-0884-x.

[13] Philippe Fournier-Viger, Cheng-Wei Wu, Antonio Gomariz, and Vincent S. Tseng. VMSP: Efficient vertical mining of maximal sequential patterns. In *Advances in Artificial Intelligence*, pages 83–94. Springer International Publishing, 2014. doi:10.1007/978-3-319-06483-3_8.

[14] Philippe Fournier-Viger, Antonio Gomariz, Michal Šebek, and Martin Hlosta. VGEN: Fast Vertical Mining of Sequential Generator Patterns. In *Data Warehousing and Knowledge Discovery*, pages 476–488. Springer International Publishing, 2014. doi:10.1007/978-3-319-10160-6_42.

[15] Smitha Harikumar. A Study on Educational Data Mining. *International Journal of Computer Trends and Technology*, 8(2):90–95, 2014.

[16] Alejandro Peña-Ayala. Educational data mining: A survey and a data mining-based analysis of recent works. *Expert Systems with Applications*, 41(4, Part 1):1432–1462, 2014. URL: http://www.sciencedirect.com/science/article/pii/S0957417413006635, doi:https://doi.org/10.1016/j.eswa.2013.08.042.

[17] Cristobal Romero and Sebastian Ventura. 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, Nov 2010. doi:10.1109/TSMCC.2010.2053532.

[18] Marie Bienkowski, Mingyu Feng, Barbara Means, et al. 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, 2012.

[19] Yuichiro Kobayashi. Computer-aided error analysis of L2 spoken English: A data mining approach. In *Proceedings of the Conference on Language and Technology 2014*, pages 127–134. DHA Suffa University Karachi, 2014.

[20] Donia Malekian, James Bailey, and Gregor Kennedy. Prediction of Students' Assessment Readiness in Online Learning Environments: The Sequence Matters. In *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*, LAK '20, pages 382–391, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3375462.3375468.

[21] Richard Weiss, Michael E. Locasto, and Jens Mache. A Reflective Approach to Assessing Student Performance in Cybersecurity Exercises. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, pages 597–602, New York, NY, USA, 2016. ACM. URL: http://doi.acm.org/10.1145/2839509.2844646, doi:10.1145/2839509.2844646.

[22] Richard Weiss, Franklyn Turbak, Jens Mache, and Michael E. Locasto. Cybersecurity Education and Assessment in EDURange. *IEEE Security Privacy*, 15(3):90–95, 2017. doi:10.1109/MSP.2017.54.

[23] Robert G. Abbott, Jonathan McClain, Benjamin Anderson, Kevin Nauer, Austin Silva, and Chris Forsythe. Log Analysis of Cyber Security Training Exercises. *Procedia Manufacturing*, 3:5088–5094, 2015. URL: http://www.sciencedirect.com/science/article/pii/S2351978915005247, doi:https://doi.org/10.1016/j.promfg.2015.07.523.

[24] Jonathan McClain, Austin Silva, Glory Emmanuel, Benjamin Anderson, Kevin Nauer, Robert Abbott, and Chris Forsythe. Human Performance Factors in Cyber Secu-

rity Forensic Analysis. *Procedia Manufacturing*, 3:5301–5307, 2015. URL: http://www.sciencedirect.com/science/article/pii/S2351978915006228, doi:https://doi.org/10.1016/j.promfg.2015.07.621.

[25] Jelena Mirkovic, Aashray Aggarwal, David Weinman, Paul Lepe, Jens Mache, and Richard Weiss. Using Terminal Histories to Monitor Student Progress on Hands-on Exercises. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, pages 866–872, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3328778.3366935.

[26] Masaryk University. KYPO Cyber Range Platform. https://www.kypo.cz/en/ [Online; accessed 2020-06-24].

[27] Pavel Čeleda, Jakub Čegan, Jan Vykopal, and Daniel Tovarňák. KYPO – A Platform for Cyber Defence Exercises. In *STO-MP-MSG-133: M&S Support to Operational Tasks Including War Gaming, Logistics, Cyber Defence*. NATO Science and Technology Organization, 2015. URL: https://is.muni.cz/publication/1319597/2015-NATO-MSG-133-kypo-platform-cyber-defence-exercises-paper.pdf.

[28] Jan Vykopal, Radek Oslejsek, Pavel Celeda, Martin Vizvary, and Daniel Tovarnak. KYPO Cyber Range: Design and Use Cases. In *Proceedings of the 12th International Conference on Software Technologies – Volume 1: ICSOFT*, pages 310–321. INSTICC, SciTePress, 2017. doi:10.5220/0006428203100321.

[29] Offensive Security. Kali Linux. https://www.kali.org/ [Online; accessed 2020-06-24].

[30] Adam Skrášek. Infrastruktura pro zaznamenávání příkazového řádku během kyberbezpečnostního tréninku. Bachelor thesis, Masaryk University, Faculty of Informatics, Brno, 2020. URL: https://is.muni.cz/th/cj7wu/.

[31] Offensive Security. Metasploit Unleashed. https://www.offensive-security.com/metasploit-unleashed/ [Online; accessed 2020-06-24].

[32] Davin McCall and Michael Kölling. A New Look at Novice Programmer Errors. *ACM Trans. Comput. Educ.*, 19(4):38:1–38:30, July 2019. URL: http://doi.acm.org/10.1145/3335814, doi:10.1145/3335814.

[33] Yu Mochizuki. Apyori. https://github.com/ymoch/apyori/ [Online; accessed 2020-06-24].

[34] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Antonio Gomariz, Ted Gueniche, Azadeh Soltani, Zhihong Deng, and Hoang Thanh Lam. The SPMF Open-Source Data Mining Library Version 2. In *Machine Learning and Knowledge Discovery in Databases*, pages 36–40. Springer International Publishing, 2016. doi:10.1007/978-3-319-46131-1_8.

[35] Philippe Fournier-Viger. SPMF: An Open-Source Data Mining Library. https://www.philippe-fournier-viger.com/spmf/ [Online; accessed 2020-06-24].

[36] Philippe Fournier-Viger. How to auto-adjust the minimum support threshold according to the data size. http://data-mining.philippe-fournier-viger.com/how-to-auto-adjust-the-minimum-support-threshold-according-to-the-data-size/ [Online; accessed 2020-06-24], May 2013.

[37] Philippe Fournier-Viger, Cheng-Wei Wu, and Vincent S. Tseng. Mining Top-K Association Rules. In *Advances in Artificial Intelligence*, pages 61–73. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-30353-1_6.

[38] Plotly Technologies Inc. Collaborative data science, 2015. URL: https://plot.ly.

[39] Michael Hahsler and Radoslaw Karpienko. Visualizing association rules in hierarchical groups. *Journal of Business Economics*, 87(3):317–335, 2017.

[40] Adam Perer and Fei Wang. Frequence: Interactive Mining and Visualization of Temporal Frequent Event Sequences. In *Proceedings of the 19th International Conference on Intelligent User Interfaces*, IUI '14, pages 153–162, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2557500.2557508.

A Content of the Thesis Archive

The thesis archive is available at https://is.muni.cz/th/cxvr2/. It contains the dataset and scripts used in our research, as well as complete results. The files of the archive are organized in the following structure:

- `dataset`: Folder containing 22 command histories from cybersecurity training in JSON/JSONL format.
- `results`
  - `charts`: Visualizations of the extracted patterns in PDF format.
  - `patterns`: Text files containing the extracted patterns in SPMF format.
  - `spmf_databases`: Text files containing databases in SPMF format created from our dataset. These files are usable as input for algorithms in the SPMF library.
- `src`
  - `cmd_log_parser.py`: A Python script for parsing the command histories.
  - `pattern_mining.py`: A Python script for mining patterns from our dataset.
  - `pattern_visualization.py`: A Python script for visualizing the mined patterns.
- `text`: Folder containing the source files of the text of this thesis.
- `LICENSE.md`: An MIT license for the source code of this thesis.
- `README.md`: A file containing information about the archive.
- `requirements.txt`: A text file describing the Python packages necessary to run the scripts.