Sequential pattern mining for ICT risk assessment and management

Michele D'Andreagiovanni[a], Fabrizio Baiardi[a], Jacopo Lipilini[a,b], Salvatore Ruggieri[a], Federico Tonelli[a,b]

[a]*Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo 3, 56127 Pisa, Italy*
[b]*Haruspex S.R.L., Via Vittorio Veneto, 126, 19124, La Spezia, Italy*

Abstract

ICT risk assessment and management relies on the analysis of data on the joint behavior of a target system and its attackers. The tools in the Haruspex suite model intelligent, goal-oriented attackers that reach their goals through sequences of attacks. The tools synthetically generate these sequences through a Monte Carlo method that runs multiple simulations of the attacker behavior. This paper presents a sequential pattern mining analysis of the attack sequence database to extract a high-level and succinct understanding of the attacker strategies against the system to assess. Such an understanding is expressed as a set of sequential patterns that cover, and possibly partition, the attack sequences. This set can be extracted in isolation, or in contrast with the behavior of other attackers. In the latter case, the patterns represent a signature of the behavior of an attacker. The dynamic tools of the suite use this signature to deploy dynamic counter-measures that reduce the security risk. We formally motivate the need for using the class of maximal sequential patterns in covering attack sequences, instead of frequent or closed sequential patterns. When contrasting the behavior of different attackers, we resort to distinguishing sequential patterns. We report an extensive experimentation on a system with 36 nodes, 6 attackers, and 600K attack sequences.

Keywords: Security risk assessment, Attack sequences, Sequential pattern mining, Maximum coverage problem

1. Introduction

Approaches to assess and manage ICT risk rely on the analysis of data on the joint behavior of the ICT system of interest, the target system, and its attackers. Most of this data is collected after the system deployment. This approach has two main drawbacks. First, it cannot assess and manage risk at design time, but only with *a-posteriori* sub-optimal remedies. This prevents a principled *by-design* approach in risk analysis and management [1]. Second, the data available for the analysis is both scarce and biased because one waits for *real* attacks to occur and they seldom include extreme events, such as low frequency - high impact attacks.

The Haruspex suite of tools [2] synthetically generate data to assess and manage ICT risk by simulating the attacker behaviors to predict how they attack the target system even before it is deployed. The suite models intelligent, goal-oriented attackers that escalate their privileges or access rights or, simply rights, by sequentially executing the elementary attacks enabled by the vulnerabilities of the target system. Each attacker aims to reach a predefined goal, i.e., to acquire a predefined set of rights. The suite applies a Monte Carlo method that runs multiple simulations of the attacker behaviors to take into account stochastic factors affecting the success or the failure of each elementary attack.

This paper proposes an innovative methodology because currently, just a few analyses of synthetic/real data exploit summaries and patterns of attack sequences to design counter-measures and deploy *dynamic* ones. This strategy matches the actual attacks against a predefined, system independent pattern to select when to deploy counter-measures for these attacks [3, 4, 5]. Instead, the proposed methodology aims at extracting a global description of each attacker strategy in building an attack sequence against the target system. This description supports both the design of counter-measures for the target system and the choice of the time to deploy dynamic counter-measures for this system. Furthermore, the description, possibly at alternative abstraction levels, is also useful to understand the “degrees of freedom” of an attacker when building its attack sequences.

This paper applies knowledge discovery approaches to extract informative patterns from attack sequences in the form of sequential patterns [6, 7, 8] where a pattern represents a high-level and declarative understanding of a collection of attack sequences. We propose to select a subset of sequential patterns that describe a number of sequences as large as possible. This is an instance of the well-known maximum coverage optimization problem [9]. The selected subsets should maximize the coverage of the sequences and also have a pairwise minimal intersection. This results in a set of independent descriptions that globally capture the whole behavior of an attacker. Also, the length of sequential patterns in the subset should be maximized because longer patterns convey more information on the abstracted sequences. We formally motivate the need for using the class of maximal sequential patterns in covering attack sequences, instead of frequent or closed sequential patterns.

The behavior of an attacker can be analyzed in isolation, or in contrast with the behavior of other attackers. In such a case, we resort to distinguishing sequential patterns [10], which are supported by the sequences of an attacker but not by those of all other ones. A *signature* of an attacker w.r.t. all the others can then be computed from such patterns. Signatures are the key elements for solving the attribution problem [11], namely for determining the identity of an attacker and then for the deployment of dynamic counter-measures. Moreover, since signatures are provided in terms of intelligible representations (sequential patterns), they can be also useful to design proper countermeasures to reduce ICT risk and to define a risk management strategy that applies the most effective counter-measure for each attacker.

We evaluate the proposed methodology with a use case of a target system with 36 nodes and 6 different types of attackers. We describe both the target system and the extensive experimentation on 600K attack sequences that has considered sequential patterns at different granularity levels (elementary attacks, nodes attacked). The trade-off between maximal covering, minimal overlap, and longest patterns is clarified by varying the minimum support threshold in sequential pattern extraction.

The paper is organized as follows. Section 2 introduces the Haruspex suite. Section 3 describes a use case including a sample ICT system and attacker modeling. Classes of sequential patterns and measures of interest for ICT risk assessment are then introduced in Section 4. Next, Section 5 defines a notion of covering for sets of sequential patterns and formally proves that maximal sequential patterns is the most appropriate class for computing covers. Section 6 discusses the covers found for attack sequences of the considered use case. Finally, we discuss related work and summarize the contribution of the paper.

S	the target system
n	a node of S
ag	an attacker
at	an elementary attack
v	a vulnerability
$v(at)$	the vulnerabilities enabling *at*
$pre(at)$	the rights to execute *at*
$res(at)$	the resources to execute *at*
$post(at)$	the rights granted if *at* succeeds
$succ(at)$	the success probability of *at*
$time(at)$	the execution time of *at*
$\lambda(ag)$	the look-ahead of *ag*

(a) List of abbreviations.

ag	**strategy**	$\boldsymbol{\lambda(ag)}$
A0	*SmartSubnetFirst*	0
A1	*maxIncr*	1
A2	*maxIncr*	2
A3	*maxProb*	1
A4	*maxProb*	2
A5	*maxAtt*	2

(b) Attacker in the use case.

Table 1

2. ICT risk assessment: the Haruspex approach

This section describes the tools of the Haruspex suite [12] that simulate the attacker behaviors against a target system to produce synthetic attack sequences. In the following, we denote by *user* the team that applies the suite tools to assess and manage the ICT risk of a *target system* that can be already working or only in its design phase. The suite builds the models of the target system and of the *attackers*, to simulate how the latter select the elementary attacks in their sequences in the scenario of interest. The accuracy of the simulation depends on the tools that build the model of the target system and those of the attackers. Table 1a defines the abbreviations we use in the following. The *builder* and the *descriptor* are the Haruspex tools that model, respectively, the target system and each attacker. Then, the *engine* uses these models to apply the Monte Carlo method and to run simulations of the behavior of a number of attackers against the target system.

2.1. Modeling an infrastructure

The *builder* is the first tool the user applies to build a model of the target system S. This model decomposes S into a network of nodes and the resulting nodes into components, i.e., hardware/software modules. Each component may be affected by some *vulnerabilities* that enable some *elementary attacks* [13, 14, 15]. Haruspex covers social engineering attacks by modeling users of S as further components with the proper vulnerabilities. If any vulnerability in $v(at)$ is effective, *at* is *enabled* and succeeds with a probability $succ(at)$, otherwise it fails. For each vulnerability v that affects a node n, the *builder* computes $pre(at)$ and $post(at)$ for each attack v enables. These properties drive how an attacker can compose elementary attacks into attack sequences. The *builder* computes $pre(at)$ and $post(at)$ by matching predefined patterns against the description of v in some *de facto* standard databases, e.g., the Common Vulnerability Enumeration [12, 13]. The information the *builder* returns in the target system model supports the simulation of any attack, provided that we know the access rights an attack grants and that we can evaluate the time to execute it and its success probability. Most collections of vulnerabilities and weaknesses describe such attack attributes. The model the *builder* returns should describe any attack sequence *ag* can implement, even if they involve distinct nodes of S. To this purpose, the user provides to the *builder* the logical topology of the interconnection structure among the nodes of S and the components of S that controls it, e.g., firewalls.

2.2. Modeling attackers

The *descriptor* models each attacker *ag* starting from four properties the user defines for it:

1. the initial rights;
2. the goal(s);
3. the selection strategy;
4. the value of $\lambda(ag)$, the look-ahead.

The goal of *ag* is the set of rights that *ag* aims to reach. We can assume that these rights result in the control of the target node n_f. An attacker that reaches a goal results in an *impact*, i.e., a loss for the owner of S [16]. *ag* sequentially selects and executes elementary attacks. *ag* can attempt an elementary attack *at* if it can access the resources in $res(at)$ and it owns the rights in $pre(at)$, either as initial rights or as rights granted by the previous attacks *ag* has successfully executed. $succ(at)$ determines the success or the failure of the execution of *at*. An attack sequence is *successful* if it reaches n_f.

Each *ag* is paired with a selection strategy that ranks the next attacks *ag* may execute according to its goals, its current set of rights and its preferences. $\lambda(ag)$ defines the look-ahead of *ag*, i.e., the largest number of attacks *ag* ranks to select those to extend its sequence. The strategy always selects one of the sequences leading to a goal, if it exists. Otherwise, the strategy ranks sequences according to attack attributes only. This may select a sequence with useless attacks, i.e., they grant rights that *ag* does not need to reach a goal. In the current version of the suite, the user pairs *ag* with one of the following strategies:

1. *maxProb*: returns the sequence with the best success probability,
2. *maxIncr*: returns the sequence granting the largest set of rights,
3. *maxEff* : returns the sequence with the best ratio between success probability and execution time of attacks,
4. *maxAtt* : returns the sequence granting the rights that enable the execution of the largest number of attacks,
5. *SmartSubnetFirst*: returns any attack that *ag* may execute and that increases these rights. It prefers attacks that enable *ag* to enter another subnetwork.

If $\lambda(ag) = 0$, then *ag* can only adopt the *SmartSubnetFirst* strategy that prefers the attacks that enable *ag* to enter a distinct subnetwork. The strategy is *Smart* because it returns attacks that *ag* can execute.

Haruspex models both the time to collect information to select the attacks and the one to implement them. Larger values of $\lambda(ag)$ result in a more accurate selection that may avoid useless attacks at the expense of the longer time *ag* spends to acquire information on the target system.

2.3. Simulation engine

The *engine* uses the model of S and those of the attackers to apply the Monte Carlo method. It executes an experiment with a large number of independent runs that simulate, for a maximum time period, the attacks of a set of attackers and the discovery of potential vulnerabilities. To guarantee run independence, the *engine* re-initializes the state of S and of any attacker before starting a new run. An experiment ends either after executing the specified number of runs or when a predefined statistic reaches the required confidence level. Each experiment returns a database of attack sequences collected in the runs.

In each time step, the *engine* considers each idle attacker *ag* that still has to reach a goal and it considers the current set of rights of *ag*. Then, *engine* computes the sequences with, at most, $\lambda(ag)$ attacks that *ag* can implement and it applies *ag* selection strategy. Then, it simulates *at*, the first attack of the sequence the strategy returns and increase the simulated time of $time(at)$ plus the selection time. As previously mentioned, the attributes the *engine* needs to simulate *at* are $succ(ag)$ and $time(ag)$ because the simulation neglects any implementation detail. If *at* succeeds and *ag* has reached a goal, the *engine* updates the corresponding impact too.

2.4. Suite validation

The overall accuracy of the Haruspex suite in forecasting the behavior of attackers against a target system has been validated by its adoption in some cyber-defense exercises [17]. In these exercises, a defender team has applied the suite before a team of human attackers, the red team, actually targeted the system. As an example, one defender team has applied the suite in two editions of the NATO Locked Shield exercise[1] to rank the software patches to apply in the short time window before the red team started its attacks. In other validation exercises, the suite has accurately predicted the attack sequences implemented by a penetration tester.

3. Use case

This section describes a use case that will be the subject of sequential pattern mining analysis later on. Fig. 1 shows the topology of the target system network. It consists of 36 nodes (hosts) plus routers and switches (that do not belong to the model). The node $n_e = 0$ is the entry node, i.e., all attack sequences start from it, and the node 34 is the goal node, i.e., $n_f = 34$. An attack sequence is successful if the attacker acquires the control of n_f.

The system vulnerabilities result in 2,859 elementary attacks, belonging to 192 attack types. Attack types group elementary attacks according to the CVE [12, 13] vulnerability they exploit. An elementary attack *at* is specific to a vulnerability at a node, i.e., instances of the same attacks that target distinct nodes are counted as distinct elementary attacks.

Example 3.1. *An example of an elementary attack that targets distinct nodes is the one that grants administrative access rights on the software HP Data Protector at node 1. The attack type is: HP Data Protector Remote Command Execution. An attack of this type that targets another node is a distinct elementary attack. As another example, an attacker may exploit an attack of type MS12020: Vulnerabilities in Remote Desktop Could Allow Remote Code Execution (2671387) (uncredentialed check) to remotely execute arbitrary code. As a third example, the Unencrypted Telnet Server type describes an attack to steal sensitive information through the Telnet protocol.*

We assume that the use case aims to assess the ICT risk due to 6 attackers A0-A5. Table 1b lists their selection strategies and look-aheads.

We have selected this use case because it is a challenging test for the discovery of attack patterns because all the attackers share the same attack surface, that of node $n_e = 0$, and they aim to control the same resource on node $n_f = 34$. Furthermore, due to both the large number of elementary attacks and the topology of the system network, each attacker can implement a large

[1] https://ccdcoe.org/event/cyber-defence-exercises.html

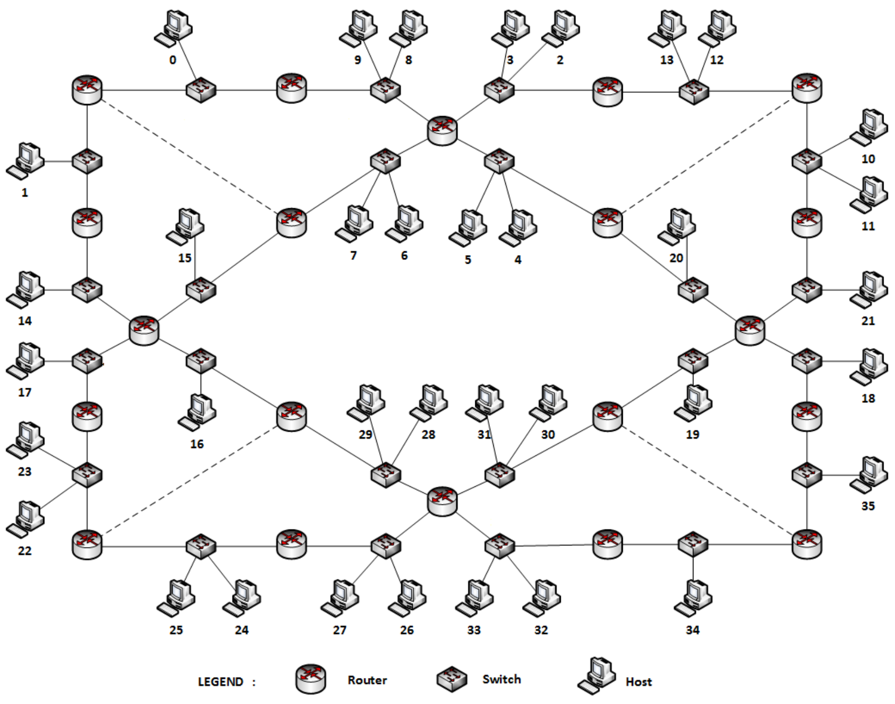

Figure 1: Topology of the Use Case.

number of sequences. From this perspective, this is a worst case for the algorithms we discuss in the following.

The *engine* returns a database with 100K successful attack sequences for each attacker. Data collected for each run includes the nodes the attacker has targeted, each attack attempted and whether it succeeded, the time of each attack, plus other information this paper neglects. Attack sequences have a length between 5 and 144, for a total of 23.7M elementary attacks in the sequences the attackers select in the simulations. The attackers have actually selected only 435 of the 2,859 possible elementary attacks. This may be due both to their selection strategies and to the lack of access rights.

Example 3.2. *A successful attack sequences involves the following nodes: 1, 9, 6, 5, 2, 20, 21, 18, 35, 34 of the target system in Fig. 1. Initially, this sequence implements an attack of type HP Data Protector < A.06.20 Multiple Vulnerabilities against node 1. Then, it exploits the attack type HP Data Protector Remote Command Execution to gain administrative access rights on nodes 9, 6, 5, and 2. Starting from these nodes, the attacker exploits further attack types, e.g., MS12020: Vulnerabilities in Remote Desktop Could Allow Remote Code Execution (2671387) (uncredentialed check) and HP System Management Homepage < 7.0 Multiple Vulnerabilities. In the end, the attacker runs an attack with type Unencrypted Telnet Server against the node 34.*

4. Sequential pattern mining over attack sequences

This section recalls the framework of sequential pattern mining with reference to the classes of patterns that will be useful for modeling attack sequences and for reasoning over them. We refer the reader to [6, 7, 8] for surveys on sequential pattern mining theory and applications.

Regarding software tools in the open source domain, a large suite of algorithms for extracting various classes of sequential patterns (frequent, closed, maximal) is the SPMF[2] Java library, described in [18]. A second tool that we will use in our experimentation is the jConSGapMiner[3] multi-core Java library [10] for extracting minimal distinguishing sequential patterns.

4.1. Sequence databases

Let $I = \{i_1, \ldots, i_m\}$ be a fixed set of *items*, where $m > 0$. Items can be symbols or, simply, natural numbers. A *sequence* $S = \langle s_1, \ldots, s_l \rangle$ is an ordered list of items, i.e., $s_i \in I$ for $i = 1, \ldots, l$. We write $l(S) = l$ to denote the length $l \geq 0$ of S. A sequence $S_1 = \langle s_1, \ldots, s_l \rangle$ is a *sub-sequence* of $S_2 = \langle r_1, \ldots, r_k \rangle$ if there exists $1 \leq \pi_1 < \ldots < \pi_l \leq k$ such that $s_1 = r_{\pi_1}, \ldots, s_l = r_{\pi_l}$. In such a case, we write $S_1 \sqsubseteq S_2$. Also, we say that S_2 *contains* S_1. We write $S_1 \sqsubset S_2$ when $S_1 \sqsubseteq S_2$ and $S_1 \neq S_2$, and say that S_2 *strictly contains* S_2. A *sequence database* $\mathcal{D} = \{S_1, S_2, \ldots, S_n\}$ is a multiset of sequences. $|\mathcal{D}| = n$ is the size of $\mathcal{D}$.

The assessment of the use case can model attack sequences by considering items at different granularities. We will consider two scenarios among the possible ones.

Definition 4.1. *In scenario $\mathcal{AS}$ (for $\mathcal{A}$ttack $\mathcal{S}$cenario), items are elementary attacks at, and sequences are ordered lists of successful elementary attacks.*

Example 4.1. *The attack sequence of Example 3.2 is modeled in scenario $\mathcal{AS}$ as $\langle at_1, at_9, at_6, at_5, at_2, at_{20}, at_{21}, at_{18}, at_{35}, at_{34} \rangle$, where at_j is some successful elementary attack attempted at node n_j. E.g., at_9 is the "HP Data Protector Remote Command Execution" at node n_9.*

Definition 4.2. *In scenario $\mathcal{NS}$ (for $\mathcal{N}$ode $\mathcal{S}$cenario), items are nodes n, and sequences are ordered lists of consecutively distinct nodes at which a successful attack took place.*

Example 4.2. *The sequence of successful attacks in Example 3.2 is modeled in scenario $\mathcal{NS}$ as $\langle n_1, n_9, n_6, n_5, n_2, n_{20}, n_{21}, n_{18}, n_{35}, n_{34} \rangle$, where at least one attack occurs in each node.*

4.2. Sequential patterns and measures of interest

Sequences may abstract subsets of $\mathcal{D}$. We denote such abstractions as sequential patterns to differentiate them from sequences in $\mathcal{D}$. Formally, a *sequential pattern SP* is a sequence. The *cover* of a sequential pattern *SP* is the set of sequences in $\mathcal{D}$ that contain *SP*. In symbols, $cover_{\mathcal{D}}(SP) = \{S \in \mathcal{D} \mid SP \sqsubseteq S\}$. The goal of *sequential pattern mining* is to extract from a sequence database a set of interesting sequential patterns. An objective measure of interest is support. The *absolute support* (or, simply, support) of *SP* is the size of its cover: $supp_{\mathcal{D}}(SP) = |cover_{\mathcal{D}}(SP)|$. The *relative support* of *SP* is the percentage of its cover over $\mathcal{D}$. We omit the subscript $\mathcal{D}$ when clear from the context.

[2]http://www.philippe-fournier-viger.com/spmf/
[3]https://github.com/slide-lig/jConSGapMiner

Example 4.3. *Consider scenario $\mathcal{AS}$. An example of a sequential pattern is $SP = \langle at_9, at_2, at_{21} \rangle$. This pattern covers all the sequences that include the attack at_9 (which is, say, at node n_9), and after zero or more steps attack at_2 (at node n_2), and after zero or more steps attack at_{21} (at node n_{21}). The sample sequence of Example 4.1 is in the cover of SP.*

Besides support, the ranking of a collection of sequential patterns may apply other subjective or domain-dependent measures of interest. The following may be specifically defined in the context of risk assessment:

- *Length* $l(SP)$: the longer a pattern is the more information it provides on a (possibly, successful) sequence of attacks of an attacker.
- *Score* $score(SP) = supp(SP) \cdot 2^{l(SP)-1}$: is a combination of support and length, where length is exponentially weighed;
- *Success distance* $success(SP)$: for a sequence S, the success distance is defined as the number of elements in S between the last one matching SP and the last in S. The success distance $success(SP)$ is the mean success distance over the sequences in the cover of SP. Intuitively, the smaller the success distance, the higher the probability that an ongoing sequence matching the pattern will become successful [4] (the attacker reach its goal).
- *Right distance* $right(SP)$: for a sequence S, the right distance is defined as the number of new rights in S acquired between the last element matching SP and the last element in S. The right distance $right(SP)$ is the mean right distance over the sequences in the cover of SP. This is a refinement of success distance that considers the number of missing rights before the success instead of the number of steps before the attacker is successful.

For example, the sequence of Example 4.1 contributes to the success distance of SP in Example 4.3 with a value of 3 because it attempts 3 more attacks after the last match with SP, namely after at_{21}.

4.3. Classes of sequential patterns

While a measure of interest can prompt useful patterns from an initial collection, the number of all possible sequential patterns is exponentially large (in the number of items and pattern length). It is then important to concentrate on specific collections of sequential patterns. The pattern mining literature has proposed a large number of collections and efficient (in some cases, parallel) algorithms to extract them from sequence databases.

Consider a fixed minimum support threshold $minsup > 0$. A sequential pattern SP is *frequent* if $supp(SP) \geq minsup$. We denote by $\mathcal{FP}_{\mathcal{D}}$ the set of frequent sequential patterns for a database of sequences $\mathcal{D}$ – and write $\mathcal{FP}$ when $\mathcal{D}$ is clear from the context. Since two frequent sequential patterns may actually denote the same set of sequences, i.e., they have a same cover, restricting to patterns with distinct covers is a lossless strategy to reduce the number of extracted patterns and to avoid duplicate analyses. SP is *closed* if it is frequent and not strictly contained in a sequential

[4]One may also consider a dual measure, the average number of elements (in a sequence of the cover of SP) up to the first matching SP. This would apparently measure how early SP can be detected in a sequence S. Unfortunately, we are sure that SP occurs in S only when S matches the last element of SP. *A fortiori*, there is no guarantee that the value of the measure applies to S, e.g.,to quantify the risk of S.

pattern SP' with the same support, i.e., $supp(SP) \geq minsup \wedge \nexists SP'. (SP \sqsubset SP' \wedge supp(SP') = supp(SP))$. We denote by $\mathcal{CP}$ the set of closed sequential patterns. Closed sequential patterns are the longest sequential patterns in the class of equivalence of patterns with a same cover. A further condensed representation consists of restricting to the longest frequent sequential patterns only. SP is *maximal* if it is frequent and not strictly contained in another frequent sequential pattern SP' i.e., $supp(SP) \geq minsup \wedge \nexists SP'. (SP \sqsubset SP' \wedge supp(SP') \geq minsup)$. We denote by $\mathcal{MP}$ the set of maximal sequential patterns. Maximal sequential patterns are closed, but the converse does not necessarily hold. In summary, $\mathcal{FP} \supseteq \mathcal{CP} \supseteq \mathcal{MP}$. For non-trivial sequence databases, the inclusions are strict.

Next lemma is referred to as the *anti-monotonicity* property of *cover* (or, equivalently, of *supp*). It is a trivial consequence of transitivity of relation $\sqsubseteq$.

Lemma 4.1. *If $SP_1 \sqsubseteq SP_2$ then $cover(SP_1) \supseteq cover(SP_2)$.*

The converse does not hold, not even for maximal sequential patterns.

Example 4.4. *Consider the sequence database $\{\langle b, c, a\rangle, \langle b, a, c\rangle\}$ and fix $minsup = 2$. The two sequential patterns $\langle a\rangle$ and $\langle b, c\rangle$ are both maximal and they both cover the whole database. However, they are incomparable w.r.t. the $\sqsubseteq$ relation.*

The pattern mining literature abounds of other methods for restricting the set of sequential patterns subject to further analysis. They are typically expressed in the form of constraints on the pattern measures or on the pattern cover. Examples of the former include lower/upper bounds on pattern length (or, on any other measure of interest), or necessarily appearing/excluded items, etc. Examples of the latter type include a maximum gap g, i.e., such that any two consecutive items in SP occur in any sequence of $cover(SP)$ within a maximum distance of g. The following other notions will be fundamental for our purposes.

Definition 4.3. *Let $\hat{\mathcal{D}}$ be a sequence database, in addition to $\mathcal{D}$. A sequential pattern SP is distinguishing w.r.t. $\hat{\mathcal{D}}$ if: (1) it is frequent, i.e., $supp_{\mathcal{D}}(SP) \geq minsup$; (2) and, it has empty cover in $\hat{\mathcal{D}}$, i.e., $supp_{\hat{\mathcal{D}}}(SP) = 0$.*

SP is minimal distinguishing w.r.t. $\hat{\mathcal{D}}$ if it is distinguishing and no $SP' \sqsubset SP$ is distinguishing.

By Lemma 4.1, minimality means that there is no SP' more general than SP, in terms of cover, which satisfies conditions both (1) and (2) .

The original definition of [10] is more general than Definition 4.3. Condition (2) is relaxed to considering infrequency w.r.t. $\hat{\mathcal{D}}$, i.e., $supp_{\hat{\mathcal{D}}}(SP) \leq maxsup$ for a threshold *maxsup*. Moreover, a maximum gap threshold g is also considered. However, we will not make use of the fully general notion in this paper. We denote by $\mathcal{DP}_{\hat{\mathcal{D}}}$ the set of distinguishing sequential patterns w.r.t. $\hat{\mathcal{D}}$ – and omit $\hat{\mathcal{D}}$ when clear from the context, and by $\mathcal{MD}_{\hat{\mathcal{D}}}$ the set of minimal distinguishing sequential patterns. It turns out that $\mathcal{FP} \supseteq \mathcal{DP} \supseteq \mathcal{MD}$.

We introduce here maximal distinguishing sequential patterns. They represent the longest frequent sequential patterns that have empty cover in $\hat{\mathcal{D}}$.

Definition 4.4. *The set of maximal distinguishing sequential patterns is $\mathcal{MDP}_{\hat{\mathcal{D}}} = \mathcal{MP} \cap \mathcal{DP}_{\hat{\mathcal{D}}}$.*

The following lemma provides us with a means of computing maximal distinguishing patterns from maximal sequential patterns and minimal distinguishing, i.e., without having to look at the database $\hat{\mathcal{D}}$ nor having to build the larger set of distinguishing patterns $\mathcal{DP}_{\hat{\mathcal{D}}}$.

Lemma 4.2. $\mathcal{MDP}_{\hat{\mathcal{D}}} = \{SP \in \mathcal{MP} \mid \exists\, SP' \in \mathcal{MD}_{\hat{\mathcal{D}}}.\ SP' \sqsubseteq SP\}$.

Proof. $\supseteq$-part). Let $SP \in \mathcal{MP}$ and $SP' \in \mathcal{MD}_{\hat{\mathcal{D}}}$ such that $SP' \sqsubseteq SP$. By Lemma 4.1, $\emptyset = cover_{\hat{\mathcal{D}}}(SP') \supseteq cover_{\hat{\mathcal{D}}}(SP)$, and then $SP \in \mathcal{MP} \cap \mathcal{DP}_{\hat{\mathcal{D}}} = \mathcal{MDP}_{\hat{\mathcal{D}}}$.

$\subseteq$-part). Let $SP \in \mathcal{MP} \cap \mathcal{DP}_{\hat{\mathcal{D}}}$. Since SP is distinguishing, some sub-sequence (at least one) $SP' \sqsubseteq SP$ of it will be minimal. Hence, the conclusion. □

5. The covering problem

5.1. Motivation and applications to risk management

Sequential patterns provide a useful abstraction for understanding attackers strategies to reach a goal. Ranking patterns using one of the measures of interest from Section 4.2 prompts subsequences that occur with high frequency among the successful ones (using support as measure), that are characterized in higher detail (using length), or that are close to success (using success/right distance). In particular, this last ranking is fundamental to design *dynamic countermeasures*, such as blocking an attacker that is following a pattern of successful attack before it reaches its goal. In this paper, however, we concentrate on a different approach, which tries and overcome the main limitation of (sequential) patterns, namely the *local view* each of them provides. A *global understanding* of the behavior of attackers is missing, and we will provide it through the covering problem described in the next subsection. First, we aim at grasping a set of strategies that characterize an attacker. Second, we aim at extracting a set of strategies that are also specific of an attacker because they are not used by the other ones. In this sense, they define an attacker *signature*. Such two sets of strategies provide an understanding of how exposed is the target system to the attacker strategies, possibly in contrast to other attackers. This understanding supports what-if analyses and the definition of a strategy that integrates in a cost effective way both *static* and *dynamic counter-measure*. A static countermeasure such as a network redesign or the adoption of software patches changes the system in a permanent way and it is usually expensive. Instead, a dynamic countermeasure, such as the dynamic update of message routing or filtering rule, is effective when attacks occur. By focusing just on the attacker(s) of interest that can reach their goals, a strategy minimizes the number of counter-measures to optimize the cost to prevent and manage risk.

This results in a security by design iterative approach that integrates the Haruspex suite and sequential pattern mining. At each step, the proposed approach generates sequence databases, characterizes attacker strategies, and designs counter-measures. Then, it starts a new step on the revised system. This iterative process takes into account the adaptive nature of intelligent attackers that can select distinct attacks as a reaction to the deployment of a counter-measure.

When static counter-measures are unavailable or too expensive, the signature of an attacker is an input to the tools that fire the deployment of dynamic counter-measures [19]. Suppose that the intrusion detection subsystem signals an attack sequence that matches the signature of an attacker that aims to control a node n_f. As soon as the signature is matched, the target system may deploy a counter-measure that filters out all the communication to n_f to isolate it from the other nodes. This expensive counter-measure will be effective only for the time to deploy a more cost-effective one. As an example, this counter-measure may only filter out the communications to n_f from the nodes the attacker uses. Lastly, the signature of an attacker is a critical input for *attack attribution*, namely for the discovery of who is attacking the target system and of its final goals. This is fundamental knowledge for enabling a strategic analysis of an attack against a critical infrastructure where the attackers will do their best to hide their identities.

Algorithm 1 greedy-kcover($k, \mathcal{P}$).

```
1: $C \leftarrow \emptyset, \mathcal{D}' \leftarrow \mathcal{D}, \mathcal{P}' \leftarrow \mathcal{P}$
2: while $|C| < k$ and $\mathcal{D}' \neq \emptyset$ and $\mathcal{P}' \neq \emptyset$ do
3:     $X \leftarrow argmax_{X \in \mathcal{P}} supp_{\mathcal{D}'}(X)$
4:     $C \leftarrow C \cup \{X\}$
5:     $\mathcal{P}' \leftarrow \mathcal{P}' \setminus \{X\}$
6:     $\mathcal{D}' \leftarrow \mathcal{D}' \setminus cover_{\mathcal{D}'}(X)$
7: end while
8: return $C$
```

5.2. The covering problem

Let us extend the *cover* notation to sets of sequential patterns.

Definition 5.1. *For a set C of sequential patterns, we define: $cover_{\mathcal{D}}(C) = \cup_{SP \in C} cover_{\mathcal{D}}(SP)$.*

We say that C *covers* the sequence database $\mathcal{D}$ if every sequence in $\mathcal{D}$ is covered by some pattern in C. The problem of finding a set C such that $cover_{\mathcal{D}}(C) = \mathcal{D}$ is an instance of the well-known *set cover problem* [9]. Typically, however, one aims at finding a set C with maximum size k (called *budget*). In the use case, for instance, domain experts may have to examine the set C. Hence, it needs to be sufficiently small for human inspection. Other constraints may arise due to the adoption of C as an input to deploy dynamic counter-measures. Since each attack is matched in real-time against each element of C, a too large number of elements may result in an unacceptable performance of the matching and, even worse, in long delays before the deployment of a counter-measure. A cover of size k may not exist, hence one aims at finding a set C of size at most k which maximizes $|cover_{\mathcal{D}}(C)|$ – the number of covered sequences from $\mathcal{D}$. This is an instance of the *maximum coverage problem* [9]. As a final generalization, we assume that sequential patterns in C are constrained to belong to a pre-determined set $\mathcal{P}$, e.g., frequent, closed, maximal, or distinguishing sequential patterns.

Definition 5.2. *Let $\mathcal{P}$ be a set of sequential patterns, and $k \geq 1$. We define the k-cover of $\mathcal{P}$:* $\mathbf{Cov}(k, \mathcal{P}) = argmax_{C \subseteq \mathcal{P} \wedge |C| \leq k} |cover(C)|$.

More than one subset $C \subseteq \mathcal{P}$ can maximize the objective function. Hence, we write $C \in \mathbf{Cov}(k, \mathcal{P})$ to denote any such subset. In addition to being a k-cover, a desirable property of C is that the sequential patterns in it overlap as little as possible – ideally, the covers of patterns in C partition the database of sequences. We measure such a property with the mean size of pairwise intersections of pattern covers.

Definition 5.3. *For a set C, we define:*

$$overlap(C) = \frac{1}{|C| \cdot (|C| - 1)} \cdot \sum_{SP_1, SP_2 \in C \wedge SP_1 \neq SP_2} |cover(SP_1) \cap cover(SP_2)|.$$

Computing the k-cover is an NP-hard problem in general [9]. A greedy algorithm chooses sequential patterns from $\mathcal{P}$ as follows: at each stage, choose the one whose cover contains the largest number of sequences uncovered by the already chosen elements. Such a greedy algorithm achieves an approximation ratio of ≈ 0.632 w.r.t. the optimal cover [9]. Algorithm 1 instantiates the greedy algorithm in the context of sequential patterns. Our implementation of this algorithm adopts a priority queue for the efficient computation of patterns with maximum support (line 3), and compressed bitmaps[5] for storing and updating covers of patterns (line 6).

We will now formally show that frequent $\mathcal{FP}$ and closed $\mathcal{CP}$ sequential patterns are not suitable choices for $\mathcal{P}$ in the computation of a cover, while maximal $\mathcal{MP}$ and minimal distinguishing $\mathcal{MD}$ sequential patterns are. First, we introduce a notation for the minimal elements of a set of patterns w.r.t. the $\sqsubseteq$ relation.

Definition 5.4. $Min(\mathcal{P}) = \{SP \in \mathcal{P} \mid \nexists SP' \in \mathcal{P}.\ SP' \sqsubset SP\}$.

For instance, we have that $Min(\mathcal{DP}) = \mathcal{MD}$, i.e., the minimal elements of distinguishing sequential patterns are the minimal distinguishing ones. The next key result shows that when looking for the k-cover of $\mathcal{P}$, one can restrict to select only minimal elements from $\mathcal{P}$.

Theorem 5.1. *Let $C \in \mathbf{Cov}(k, \mathcal{P})$. There exists $C' \in \mathbf{Cov}(k, Min(\mathcal{P}))$ such that $cover(C) = cover(C')$ and $|C| \geq |C'|$.*

Proof. We define C' as follows. For every $SP \in C$ choose any $SP' \in Min(\mathcal{P})$ such that $SP' \sqsubseteq SP$. For a given SP, at least one such a SP' exists – it could be SP itself, if it is minimal in $\mathcal{P}$. Any of such SP''s can be chosen. By construction $|C'| \leq |C| \leq k$ (in fact, a same minimal element can be chosen for two SP's in C). By Lemma 4.1, $cover(SP) \subseteq cover(SP')$. As a consequence:

$$cover(C) = \cup_{SP \in C} cover(SP) \subseteq \cup_{SP' \in C'} cover(SP') = cover(C').$$

Since C is a k-cover and $\mathcal{P} \supseteq Min(\mathcal{P})$, then $|cover(C)|$ is maximal, which implies that the inclusion above is an equality. Finally, observe that a k-cover from $\mathbf{Cov}(k, Min(\mathcal{P}))$ cannot cover more sequences than a k-cover of $\mathcal{P}$, again because $\mathcal{P} \supseteq Min(\mathcal{P})$. Since $C' \subseteq Min(\mathcal{P})$ and $|C'| \leq k$ and $|cover(C')|$ is maximal, we conclude $C' \in \mathbf{Cov}(k, Min(\mathcal{P}))$. □

The greedy algorithm, in particular, returns sequential patterns from $Min(\mathcal{P})$.

Lemma 5.1. *Assume that Algorithm 1 selects at line 3 an X which maximizes $supp_{\mathcal{D}'}(X)$ and is minimal. Then **greedy-kcover(k, $\mathcal{P}$)** $\subseteq Min(\mathcal{P})$.*

The consequence of Theorem 5.1 is considerable. It makes no sense to try and cover a database of sequences using frequent patterns or closed patterns, as shown in the next examples.

Example 5.1. *It is readily checked that the minimal element of frequent sequential patterns $\mathcal{FP}$ is the empty sequence, namely $Min(\mathcal{FP}) = \{\langle\rangle\}$. Trivially, $\{\langle\rangle\}$ is a cover of any sequence database. By Theorem 5.1, $\{\langle\rangle\} \in \mathbf{Cov}(k, \mathcal{FP})$. Unfortunately, the empty sequential pattern is of no practical help in any application.*

Even when restricting to sets of patterns without the empty one, the problem still persists, making sequential patterns unuseful because a cover consists of singleton sequences.

Example 5.2. *Consider non-empty frequent sequential patterns. We have that $Min(\mathcal{FP} \setminus \{\langle\rangle\}) \subseteq \{SP \mid \exists i \in I.\ SP = \langle\{i\}\rangle \wedge SP \in \mathcal{FP}\}$. Then, to find a cover, we can consider only sequential patterns with one item. In other words, we can avoid extracting sequential patterns at all.*

[5]http://roaringbitmap.org

For closed sequential patterns $\mathcal{CP}$, one can reason as in the above example, and reach similar conclusions[6]. To find a way out, one may consider $\mathbf{Cov}(k, \mathcal{P})$ for $\mathcal{P}$ being a set of frequent or closed sequential patterns with support in a range $[minsup, maxsup]$. However, which maximal threshold *maxsup* to choose remains unclear. We propose, instead, to use maximal $\mathcal{MP}$ and maximal distinguishing sequential patterns $\mathcal{MDP}$, for three reasons. First, we have already observed that the longer a sequence of attacks is the better it describes the behavior of the attacker on a specific target system. Second, such classes (and the minimal distinguishing sequential patterns) are closed under minimality.

Lemma 5.2. *$Min(\mathcal{MP}) = \mathcal{MP}$, $Min(\mathcal{MD}) = \mathcal{MD}$, and $Min(\mathcal{MDP}) = \mathcal{MDP}$.*

The drawbacks highlighted for frequent and closed sequential patterns are then overcome. As the third reason, albeit a k-cover $C \subseteq \mathcal{MP}$ is not a partition of $\mathcal{D}$, for any two sequential patterns in C the number of sequences they share can be upper-bounded. This is a useful property when a business action has to be done for each sequential pattern in a cover, but sequences should not be subject to several business actions. In particular, in our use case we aim at finding a cover consisting of sequential patterns which abstract attack strategies alternative to each other, hence with minimal overlap (see Definition 5.3).

Lemma 5.3. *For $SP_1, SP_2 \in \mathcal{MP}$ with $SP_1 \neq SP_2$: $|cover(SP_1) \cap cover(SP_2)| < 2 \cdot minsup - 1$.*

Proof. Let SP be the longest initial sub-sequence shared by SP_1 and SP_2. $SP \sqsubset SP_1$, since $SP = SP_1$ and $SP_1 \neq SP_2$ would imply that $SP_1 \sqsubset SP_2$, and then SP_1 would not be maximal. Analogously, we have $SP \sqsubset SP_2$. Thus, let $SP_1 = SP \cdot \langle i \rangle \cdot \hat{SP}_1$ and $SP_2 = SP \cdot \langle j \rangle \cdot \hat{SP}_2$, where $\cdot$ is the sequence appending operator, and $i \neq j$. Any $S \in cover(SP_1) \cap cover(SP_2)$ satisfies at least one of the following conditions: (1) $SP \cdot \langle j, i \rangle \cdot \hat{SP}_1 \sqsubseteq S$; or (2) $SP \cdot \langle i, j \rangle \cdot \hat{SP}_2 \sqsubseteq S$, depending on whether j appears before i in S or vice-versa. The number of sequences n_1 that satisfy (1) must be smaller than *minsup*, i.e., $minsup - 1 \geq n_1$, otherwise SP_1 would not be maximal since $SP_1 \sqsubset SP \cdot \langle j, i \rangle \cdot \hat{SP}_1$. Similarly, the number n_2 of sequences that satisfy (2) is smaller than *minsup*, i.e., $minsup - 1 \geq n_2$. We conclude then: $|cover(SP_1) \cap cover(SP_2)| \leq n_1 + n_2 \leq 2 \cdot minsup - 2 < 2 \cdot minsup - 1$. □

As a consequence, a set of maximal (distinguishing) sequential patterns has a controllable bound on the overlap measure. This holds in particular for a k-cover and for the set returned by the greedy Algorithm 1.

Corollary 5.1. *Let $C \subseteq \mathcal{MP}$. We have: $overlap(C) < 2 \cdot minsup - 1$.*

5.3. Extended definitions of sequences and sequential patterns

A conservative extended definition of a sequence [20] assumes that $S = \langle A_1, \ldots, A_l \rangle$ is an ordered list of non-empty itemsets, where an *itemset* is a set of items, i.e., $\emptyset \subset A_i \subseteq I$ for $i = 1, \ldots, l$. A sequence $S_1 = \langle A_1, \ldots, A_l \rangle$ is a *sub-sequence* of $S_2 = \langle B_1, \ldots, B_k \rangle$ if there exists $1 \leq \pi_1 < \ldots < \pi_l \leq k$ such that $A_1 \subseteq B_{\pi_1}, \ldots, A_l \subseteq B_{\pi_l}$. This conservatively extends the $\sqsubseteq$ relation as well. The notions of support, sequential patterns, and the various classes (frequent,

[6]It turns out that $Min(\mathcal{CP}) = \{\langle\rangle\}$. For non-empty closed sequential patterns, it turns out $Min(\mathcal{CP} \setminus \{\langle\rangle\}) \subseteq \{[FP]_\theta \mid \exists i \in I.\ FP = \langle\{i\}\rangle \wedge SP \in \mathcal{FP}\}$, where $[FP]_\theta$ is the closed sequential pattern in the class of θ-equivalence of SP, namely all sequential patterns SP' such that $cover(SP') = cover(SP)$.

closed, maximal, distinguishing) of sequential patterns readily lift from the above definitions (see [6, 7, 8] for details). Our key result Theorem 5.1 also generalizes, since it basically exploits the partial order $\sqsubseteq$ and the anti-monotonicity Lemma 4.1. The only result that must be relaxed is Lemma 5.3, which we restate next.

Lemma 5.4. *Assume the extended definition of sequences. Let $SP_1, SP_2 \in \mathcal{MP}$ with $SP_1 \neq SP_2$. We have: $|cover(SP_1) \cap cover(SP_2)| < 3 \cdot minsup - 2$.*

Proof. Let SP be the longest initial sub-sequence shared by SP_1 and SP_2. $SP \sqsubset SP_1$, since $SP = SP_1$ and $SP_1 \neq SP_2$ would imply that $SP_1 \sqsubset SP_2$, and then SP_1 would not be maximal. Analogously, we have $SP \sqsubset SP_2$. Thus, let $SP_1 = SP \cdot \langle A \rangle \cdot \hat{SP}_1$ and $SP_2 = SP \cdot \langle B \rangle \cdot \hat{SP}_2$, where $A \neq B$ are itemsets. Any $S \in cover(SP_1) \cap cover(SP_2)$ satisfies at least one of the following conditions: (1) $SP \cdot \langle B, A \rangle \cdot \hat{SP}_1 \sqsubseteq S$; or (2) $SP \cdot \langle A, B \rangle \cdot \hat{SP}_2 \sqsubseteq S$; or (3) $SP \cdot \langle A \cup B \rangle \cdot \hat{SP}_2 \sqsubseteq S$, depending on whether B appears before A in S or vice-versa or they are included in the same sequence element (this last option is impossible if itemsets are singletons, as in Lemma 5.3). The proof then proceeds analogously to the proof of Lemma 5.3. □

The extended definition of sequences may enable the analysis of further analysis scenarios where elements of a sequence are sets of events. For instance, an extension of the $\mathcal{AS}$ scenario may consider a sequence of the sets of all successful elementary attacks at a given node. An extension of the $\mathcal{NS}$ scenario may consider a sequence of the sets of all nodes consecutively and successfully attacked by the same attack type. On the other side, however, the extended definition leads to denser datasets of sequences, and then to a larger number of sequential patterns to extract and reason about. While efficient algorithms exists for extracting (frequent and closed) sequential patterns for relatively low minimum support threshold [18, 21], sequential patterns with itemsets are slightly less intuitive to interpret for a security expert, since they do not specify an order of the events in a set. For such reasons, we formulated our framework on sequences of items.

6. Experimental analysis of the use case

This section investigates on covering a database of attack sequences using sequential patterns of different types. Maximal sequential patterns represent high-level and condensed abstractions of the attack sequences of an attacker in isolation. We will look for subsets of patterns that maximize their cover and minimize their overlap. Minimal distinguishing sequential patterns abstract attack strategies specific to an attacker and not replicated by the other ones. They represent an attacker signature with respect to the target system under consideration.

6.1. Dataset and basic statistics

Recall that the input dataset in the use case consists of 600K successful attack sequences, 100K for each of the six attackers (see Table 1b). The empirical distribution of sequence lengths is reported in Fig. 2 for both the $\mathcal{AS}$ scenario (left plot) and the $\mathcal{NS}$ scenario (center plot). The plots also report the kernel density estimation line. In both plots, we can observe a mixture of two distributions. In fact, the lengths of the sequences of A1 and A2 are either 12 or 13. The lengths of the sequences of the other attackers are well approximated by Gaussian distributions.

Fig. 2 (right) also shows the empirical distribution of support for singleton sequential patterns, namely those of the form $\langle at \rangle$ for an elementary attack at in the $\mathcal{AS}$ scenario, and of the form $\langle n \rangle$ for a node n in the $\mathcal{NS}$ scenario. Singleton sequential patterns are ordered by rank,

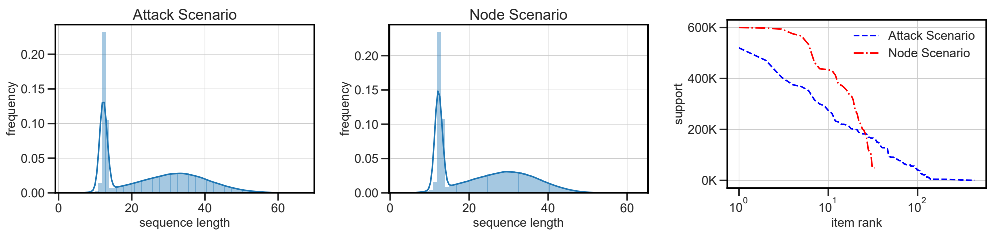

Figure 2: Distribution of sequence length (left and center), and support of singleton sequential patterns by rank (right).

from the one with the highest support to the one with the lowest one. Notice that the x-axis is in logarithmic scale. In the $\mathcal{AS}$ scenario, the empirical distribution is well-fitted by an exponential distribution. In the $\mathcal{AS}$ scenario, a few nodes occur very frequently in attack sequences. Looking at Fig. 1, such nodes include the target node $n_f = 34$ (all sequences end there), and nodes directly reachable from the entry node $n_e = 0$ (all sequences start from there).

6.2. Covering with frequent and closed sequential patterns

Let us start considering the cover of the sequences of an attacker. As shown by Theorem 5.1 and in Example 5.2, using frequent or closed sequential patterns, we end up with a max covering set consisting of the empty sequential pattern or, if excluding it, of singleton patterns only. Consider again Fig. 2 (right). For the $\mathcal{NS}$ scenario, there is a single node whose cover is the whole database of sequences of all attackers. It is the goal node, i.e., $n_f = 34$ – since we are considering attack sequences which are successful. It turns out that $\{\langle n_f \rangle\} \in \mathbf{Cov}(1, Min(\mathcal{FP}))$, i.e., the singleton $\langle n_f \rangle$ is a complete cover. Moreover, **greedy-kcover(**1**,** $\mathcal{FP}$**)** $= \{\langle n_f \rangle\}$,i.e., the singleton is returned by the greedy algorithm. However, it is of little or no utility, because: (1) first, the singleton sequential pattern $\langle n_f \rangle$ cannot enable effective dynamic counter-measures, because if the attacker is attempting an attack against n_f then it has almost reached its goal; (2) second, $\langle n_f \rangle$ gives no hint on the attacker strategies to reach the final node, e.g., on the preferred sub-networks to follow. For the $\mathcal{AS}$ scenario, we reach a similar conclusion, yet now the maximal cover consists of two elementary attacks, both occurring at node n_f.

6.3. Covering with maximal sequential patterns

Maximal sequential patterns are an appropriate class for the maximal covering problem. Consider the scenario $\mathcal{AS}$. For each attacker, we extract the maximal sequential patterns from the 100K attack sequences of the attacker, and for $minsup = 5K$, namely 5% relative minimum support threshold. Starting from the set of maximal sequential patterns, we compute the cover returned by the greedy Algorithm 1. Fig. 3 shows the fraction of the 100K sequences that $C =$ **greedy-kcover(**k**,** $\mathcal{MP}$**)** covers at the variation of k for attackers A0-A5 (left) , the mean overlap of patterns in C (center), and the average length of patterns in C (right). With the top 20 sequential patterns, we are able to cover 80% or more of the sequences of each attacker, and almost 100% for A1 and A2. With 40 sequential patterns, we reach almost 100% coverage in all cases. This means it is sufficient for a domain expert to look at $k = 40$ maximal sequential patterns for a high level condensed characterization of each attacker. The average length of those sequential patterns ranges from 3.5 to 7, and their overlap ranges from 400 to 700 for distinct attackers. A1 and A2 have the best of such values, namely a low overlap and a large length. Hence, a small

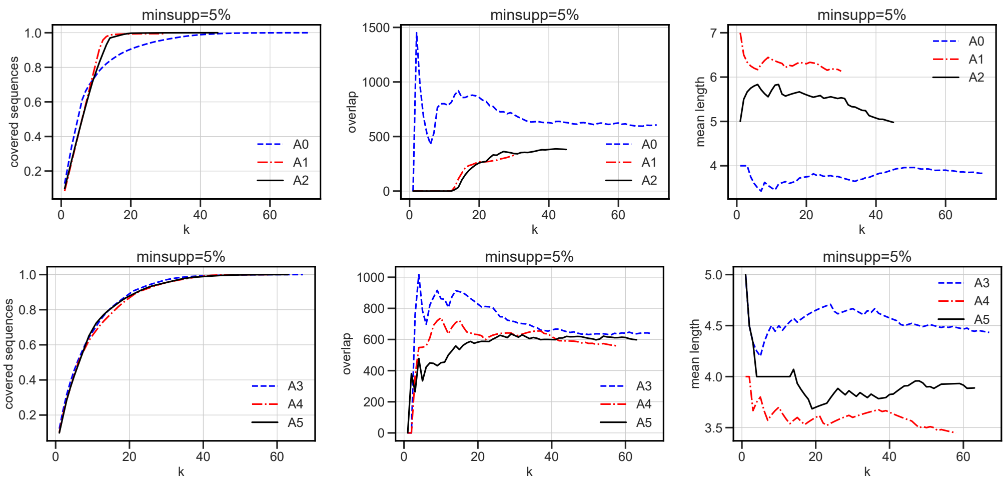

Figure 3: Scenario $\mathcal{AS}$, *minsupp* = 5%, output of **greedy-kcover(**k**,** $\mathcal{MP}$**)**. Top: A0-A2. Bottom: A3-A5.

number of sequential patterns characterizes strategies that maximize the set of acquired rights. These patterns are informative because of their non-trivial length and small overlap.

Notice that the values for the overlap are much lower than the theoretical bound of Corollary 5.1. Nevertheless, is it possible to compute a k-cover with smaller overlaps? The only parameter that can control the overlap is the minimum support threshold. Fig. 4 shows the same plots as Fig. 3, but for $minsup = 3K$, namely 3% relative minimum support. On the negative side, the plots of the covered sequences highlight that the number k needed to achieve the same fraction of coverage is now larger. E.g., to cover 80% of sequences, we need $k = 75$ patterns in the case of A0-A2, and $k = 100$ for A3-A5. On the positive side, the overlap is more than halved, which is proportionally more than the lowering of the minimum support threshold. Another positive aspects is that length of maximal sequential patterns is longer for lower support, hence they provide more information on the attacker paths over the target system. In a less formal way, the trade-off between the number of sequential patterns and their utility can be explained as follows. On the one hand, the lower the minimum support threshold, the smaller is the cover of a maximal sequential patterns, and then the larger is the number of them needed to cover a fraction of sequences. On the other hand, the lower is the minimum support threshold, the longer are maximal sequential patterns and the smaller is the overlap between any pair of maximal sequential patterns, as stated by Lemma 5.3.

Let us consider now the scenario $\mathcal{NS}$. Here the database of sequences is denser and this causes a larger number of sequential patterns to be extracted for a given minimum support. Fig. 5 shows the same plots as for scenario $\mathcal{AS}$. Attacker A1 is hard characterizable[7] for a minimum support as high as 5%. All other attackers (A3-A5 not shown in the figure) have a cover close to 100% for $k \approx 100$. The average length of sequential patterns in the cover is rather long (6 to 11). This means that those patterns are highly informative about the paths of the attackers.

As a final note, we point out that the complexity of the strategies/paths an attacker can implement increases with the number of sequential patterns to reach a certain fraction of covered

[7]There is only one sequential pattern in the output of the greedy algorithm. This is the reason for no A1 line in the plots of overlap and mean length.

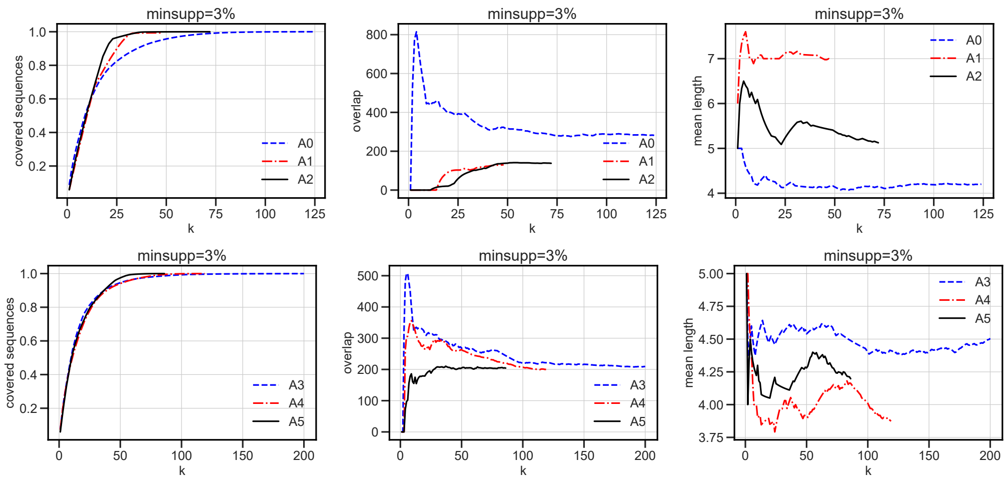

Figure 4: Scenario $\mathcal{AS}$, *minsupp* = 3%, output of **greedy-kcover**(k, $\mathcal{MP}$). Top: A0-A2. Bottom: A3-A5.

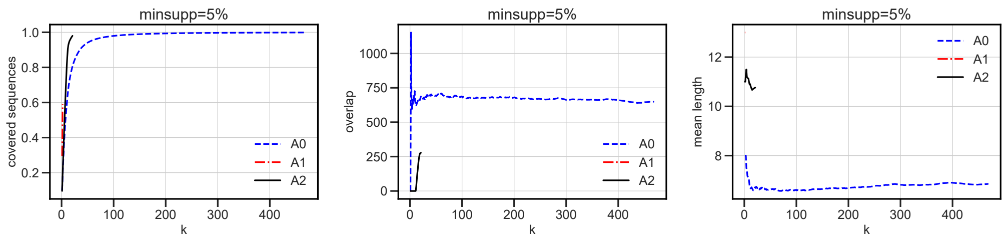

Figure 5: Scenario $\mathcal{NS}$, *minsupp* = 5%, output of **greedy-kcover**(k, $\mathcal{MP}$).

sequences. Even the number of counter-measures to deploy to stop the attacker increases with the one of sequential patterns. At one extreme, a topology with just a single path produces a single maximal sequential pattern. The left-hand side plots in Figs. 3-5 evaluate the "degree of freedom" of an attacker in building its attack sequences against the target system. Hence, we can conclude that the strategies of A3-A5 against the use case system are more complex than those of A0-A2.

6.4. *Covering with minimal/maximal distinguishing sequential patterns*

Let us consider now the computation of k-covers of distinguishing sequential patterns, where for a given attacker $\mathcal{D}$ is the database of its 100K attack sequences, and $\hat{\mathcal{D}}$ is the database of the 500K attack sequences of the remaining attackers. As discussed in Section 4.3, when computing a cover, we have two options as classes of sequential patterns: minimal distinguishing $\mathcal{MD}$, and maximal distinguishing $\mathcal{MDP}$. Fig. 6 shows the results of the former case in the $\mathcal{AS}$ scenario. Minimal distinguishing patterns can be extracted very efficiently for very low support. Here, we consider *minsupp* = 0.1%, namely 100 sequences. Even with such a low threshold, sequences of A2 require a large k for complete coverage, and those of A3 can only be covered for less than 40%. Furthermore, the overlap is noticeable larger for A0, A4, and A5 and we cannot state any theoretical bound for distinguishing sequential patterns such as the one in Corollary 5.1.

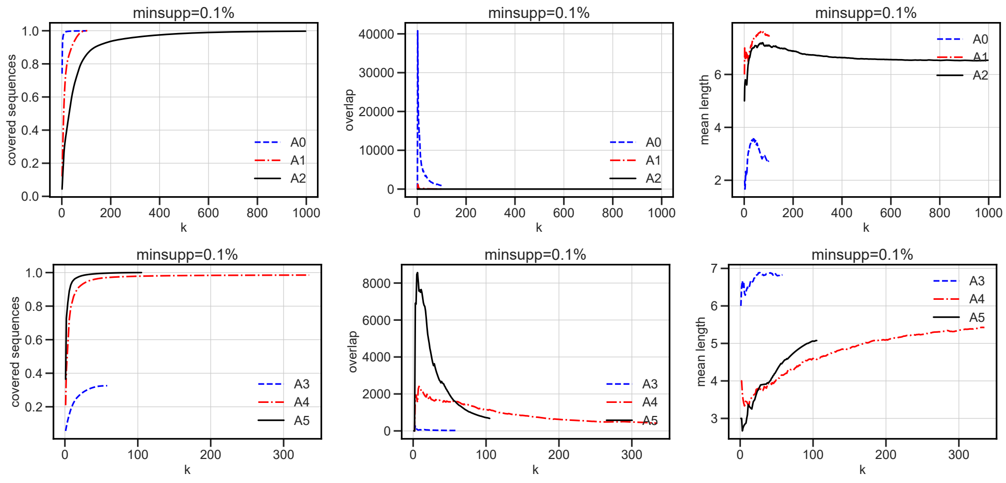

Figure 6: Scenario $\mathcal{AS}$, *minsupp* = 0.1%, output of **greedy-kcover**(k, $\mathcal{MD}$).

As anticipated, in our challenging use case, all the attackers share the attack surface, $n_e = 0$, and aim to control the same resource on node $n_f = 34$. This strongly increase the complexity of defining a signature. Anyway, even in this worst case, the proposed algorithm can discover a signature for all the attackers but A3. The lack of a signature for an attacker points out to the system designer that some attacks could not be attributed. If attribution is critical for a system, this may require the adoption of some architectural changes to simplify attribution. The proposed approach enables the system designer to evaluate how alternative changes simplify attribution before implementing them and to select the most effective one.

The cover of A0 consists of patterns that include at most four attacks, while more than half of the patterns include one attack only. Looking at the nodes that the attacks target, most attacks in the sequential patterns of the cover occur at node 5 and/or 6 (see Fig. 1). Moreover, these attacks are among the first ones in the sequences of A0. This is highly informative characterization of A0 vs the other attackers, i.e., it is a signature of A0. The patterns in the cover for A1 and A2 are longer, on average more than six attacks. This is not surprising because their selection strategies maximize the set of rights, which is monotone with the lookahead parameter. Hence, their sequences can be differentiated only at a finer grain of detail. When contrasting A3 with A4, the opposite case occurs because it is easier to cover A4 than A3. Both A3 and A4 select attacks according to their success probability. This is strongly related with the attack properties rather than with the goals of A1 and of A2. Furthermore, the lookahead of A3 is 1 while the one of A4 is 2. This implies that A4 can even select a sequence where the first attack has a low success probability because this is compensated by the success probability of the second one. This does not happen for A3. To explain the complexity of covering A3, we analyze the minimal distinguishing patterns separately for A3 vs each of the other attackers. This analysis shows that A3 is hardly distinguishable from A0 because in the target system the attacks to pass from one subnetwork to another have a larger success probability. Hence, the sequences of A3 and A0 are strongly correlated because their strategies select, for distinct reasons, similar attacks. This unexpected property of the target system explains the complexity of distinguishing the two attackers and we have discovered it as a side effect of the computation of the attacker signatures.

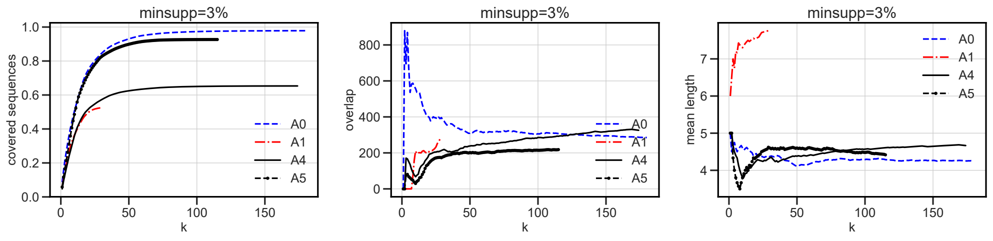

Figure 7: Scenario $\mathcal{AS}$, *minsupp* = 3%, output of **greedy-kcover**(k, $\mathcal{MDP}$).

The property is also rather unexpected and interesting for system design. In fact, usually we partition a system into subnetworks also to increase the complexity for the attacker of moving from one network to another one. This contradicts the large success of attacks to move across network in the use case system and it can be discovered only through an analysis as the proposed one.

Finally, we report in Fig. 7 the plots obtained by using maximal distinguishing sequential patterns. Since they are computed starting from maximal patterns and distinguishing patterns (see Lemma 4.2), this requires a higher minimum support threshold of 3%. A2 and A3 are absent because no cover is returned for that support level. Moreover, A1 and A5 are covered only up to about 50% and 65% respectively. On the positive side, overlap of patterns for A0 is lower and they are longer than minimal distinguishing patterns. This confirms that maximal (distinguishing) sequential patterns are an effective way of controlling those two characteristics of covers. Unfortunately, computing maximal sequential patterns for very low support is infeasible, as there is a large number of them. As a further example of this limitation, we can extract maximal distinguishing sequential patterns in the $\mathcal{NS}$ scenario with a support as low as 5% only for A0 (figure not shown).

7. Related work

Since the original works [20, 22], sequential patterns and their generalization have been studied for more than 20 years in the data mining literature [6, 7, 8]. They have been applied in distinct domains: market basket analysis [20], weblog analysis [23], malware classification [24], bioinformatics [25], operational risk [26], etc. Numerous sequential, parallel, and distributed algorithms have been proposed for extracting sequential patterns from a database of sequences [21, 27, 28], also for the extended format of Section 5.3. We used here the open source SPMF Java library for maximal sequential patterns [18], and the jConSGapMiner Java implementation for minimal distinguishing sequential patterns [10].

Sequential patterns extracted from a database of sequences are ranked on the basis of some interestingness measure, typically including support. Most of the approaches extract *all* patterns that satisfy some condition on individual interestingness measures, e.g., unexpectedness [29] or other user-defined constraints [30]. This approach raises the problem of selecting relevant sequential patterns *a-posteriori*, and it is subject to the problem of pattern explosion. A few exceptions [31, 32] consider extracting a small, non-redundant, and interesting *set* of sequential patterns, which, as a whole, satisfy a global interesting criterion – the Minimum Description Length (MDL) principle. The MDL principle identifies a set of patterns that provide the best lossless compression of the data. This is an alternative notion of "covering" from ours, since we do not require that a cover is able to reconstruct the database of sequences.

Our adaption of the maximum coverage problem and its greedy algorithm [9] to rank sequential patterns based on their residual coverage is a novel approach. A related one is the computation of *coverage patterns*, a class of non-sequential itemsets X, such that the single items in X cover a specified percentage of transactions and they overlap at most for a specified threshold [33]. Such covering and non-overlapping requirements are, however, referred to single itemsets/patterns, while we consider sets of (sequential) patterns.

Regarding the comparison of sequential patterns across two or more databases of sequences, in addition to minimal distinguishing patterns, we discuss here contrast sequential patterns and sequential pattern classification. *Contrast sequential patterns* [34] are supported in a database of sequences $\mathcal{D}$ with a minimum (contrast) ratio with respect to their support in another database $\hat{\mathcal{D}}$. In our context, a ratio of ∞ would be required, since we look for sequential patterns with zero support in $\hat{\mathcal{D}}$. In this special case, contrast sequential patterns boil down to distinguishing sequential patterns $\mathcal{DP}$. In *sequential pattern classification*, sequences in a database are labelled with a class value $C = v$. The sequential pattern classification problem consists of extracting rules $SP \rightarrow C = v$ (classifier construction), for a sequential pattern SP, and of composing those rules to predict the class value of a new sequence (classifier prediction). We refer to [35] for the state-of-the-art approach. In our case, the database of sequences consists of all 600K attack sequences, and the class of a sequence is the attacker which originated the sequence, namely $C = A_i$, for some $i \in \{0, \ldots, 5\}$. The classifier construction would then produce rules that characterize each attacker. In general, however, a rule $SP \rightarrow C = v$ does not have 100% confidence (i.e., all sequences that include SP have class label $C = v$). With distinguishing sequential patterns, we instead achieve that. Moreover, two rules may overlap, in the sense that a sequence may include both sequential patterns in the rule premises. The degree of overlap is not controllable by the user. Our approach instead provides a theoretical bound and, in practice, a very low overlap.

A large number of Intrusion Detection Systems (IDS) have adopted data-driven approaches, e.g., by learning classifiers [3], association rules [4], and closed sequential patterns [5] from event logs. All of them struggle with the lack of data to build accurate models. The adoption of the Haruspex methodology can both overcome this limitation and apply the approach at system design-time. Model-driven approaches have been also used in the context of IDS. The behavior of attackers and defenders can be modeled as a game [36], with the purpose of predicting the future actions of an attacker, optimizing counter-measures, and evaluating the security of the target system. These approaches, however, are orthogonal to our data-driven methodology.

Sequential patterns are not the only formalism that can be used to abstract sequential event logs. A related approach is process mining, whose goal is "to use event data to extract process-related information, e.g., to automatically *discover* a process model" [37]. Process models are stated in a high-level formalism such as Petri Nets or BPMN. Process mining has been applied to detecting anomalous process executions in the context of security monitoring [38]. The approach first constructs a process model from normal event logs. Then, new transaction logs are executed against the model for conformance checking. In our context, the discovery of a process model from attack logs would result in a (high-level) description of the target system S itself. In fact, the network topology and the node vulnerabilities constrain an attacker's behavior as they determine, respectively, how the attacker can move and which attacks it can execute against a node. However, the target system is a given input to our problem (see Section 2.1), not an output to be discovered. As a consequence, process mining does not appear readily applicable to the understanding of an attacker's behavior and, furthermore, to contrast behaviors of different attackers.

Finally, this paper significantly extended the preliminary results appeared in [39] along several directions. First, we characterize signatures of attackers through minimal distinguishing sequential patterns. Second, the theoretical framework for reasoning over the cover of a database of sequences has been enhanced, including now an explicit greedy covering algorithm and formal results about its properties. Third, the experimental section has been largely extended, with an extensive study on two granularity of analyses (attack and node scenarios).

8. Conclusions

We have proposed a novel data-driven approach, based on sequential pattern mining, in ICT risk assessment and management that may reduce risk at design-time. Starting from a sequence database generated through Monte Carlo simulations, we formally showed that maximal sequential patterns are preferable over frequent and closed sequential patterns for use in the maximal/greedy coverage of a database of sequences. A theoretical bound is stated on the overlap between any two maximal sequential patterns. We also proposed minimal distinguishing sequential patterns and the intersection between maximal sequential patterns and distinguishing sequential patterns as appropriate classes for covering a database of sequences without intersecting another database. This theoretical contributions hold in general. Our focus was primarily in the context of sequences of attacks to an ICT system. A non-trivial use case has shown that a greedy *k*-cover based on maximal (distinguishing) sequential patterns can provide an abstraction to understand the behavior of an attacker in isolation and to contrast this behavior against other attackers. This abstraction can help the design of counter-measures against specific attackers to reduce ICT risk for the target system at hand. Furthermore, it also simplifies the definition of a strategy to integrate static and dynamic countermeasures to minimize risk.

We outline two lines of future work. First, we struggled with efficiency limitations in computing maximal distinguishing sequential patterns starting from the sets of maximal and distinguishing ones. We adopted a post-processing approach starting from maximal and minimal distinguishing patterns (see Lemma 4.2) which suffers from the computation of a large number of maximal sequential patterns for moderately low minimum support thresholds. We will consider variants of the algorithms for the extraction of distinguishing sequential patterns [10] for directly computing maximal distinguishing sequential patterns from two databases of sequences. Furthermore, a direct extraction algorithm could also integrate constraints on other domain-dependent measures of interests (see Section 4.2). For example, success distance is a key measure for ranking covers or sequential patterns that allow for the early detection of attacks.

The second line of future work generalizes the proposed approach to multi-dimensional sequences. In our use case, we dealt with two scenarios of analysis in isolation considering, respectively, sequences of elementary attacks and sequences of attacked nodes. A multi-dimensional sequence can contain items at different levels of a hierarchy, e.g., in our case elementary attacks and attacked nodes. Multi-dimensional sequential patterns [40], or the even more general multi-dimensional relational sequential patterns [41], extend sequential patterns with items over a hierarchy. Their usage would enable mixing events of both types in the same solution, depending on how well they cover sequences. In other words, the choice of the abstraction level to consider has not to be made *a-priori*, but the right level will result from the covering algorithm.

References

[1] Joint Task Force Transformation Initiative Interagency Working Group, SP 800-30 revision 1: Guide for conducting risk assessments, National Institute of Standards & Technology, 2012.

[2] F. Baiardi, C. Telmon, D. Sgandurra, Haruspex: Simulation-driven risk analysis for complex systems, ISACA Journal 3 (2012) 46–51.

[3] W. Lee, S. J. Stolfo, K. W. Mok, Adaptive intrusion detection: A data mining approach, Artif. Intell. Rev. 14 (6) (2000) 533–567.

[4] R. Katipally, W. Gasior, X. Cui, L. Yang, Multistage attack detection system for network administrators using data mining, in: Proc. of the Cyber Security and Information Intelligence Research Workshop (CSIIRW 2010), no. 51, ACM, 2010.

[5] H. Brahmi, S. B. Yahia, Discovering multi-stage attacks using closed multi-dimensional sequential pattern mining, in: Proc. of Int. Conf. on Database and Expert Systems Applications (DEXA 2013), Vol. 8056 of LNCS, Springer, 2013, pp. 450–457.

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

[7] N. R. Mabroukeh, C. I. Ezeife, A taxonomy of sequential pattern mining algorithms, ACM Comput. Surv. 43 (1) (2010) 3:1–3:41.

[8] C. Mooney, J. F. Roddick, Sequential pattern mining - approaches and algorithms, ACM Comput. Surv. 45 (2) (2013) 19:1–19:39.

[9] D. S. Hochbaum, Approximating covering and packing problems: Set cover, vertex cover, independent set, and related problems, in: D. S. Hochbaum (Ed.), Approximation Algorithms for NP-hard Problems, PWS Publishing Co., 1997, pp. 94–143.

[10] X. Ji, J. Bailey, G. Dong, Mining minimal distinguishing subsequence patterns with gap constraints, Knowl. Inf. Syst. 11 (3) (2007) 259–286.

[11] Z. Liang, R. Sekar, Fast and automated generation of attack signatures: a basis for building self-protecting servers, in: ACM Conference on Computer and Communications Security, ACM, 2005, pp. 213–222.

[12] F. Baiardi, F. Corò, F. Tonelli, D. Sgandurra, A scenario method to automatically assess ICT risk, in: Euromicro Int. Conf. on Parallel, Distributed, and Network-Based Processing (PDP 2014), IEEE Computer Society, 2014, pp. 544–551.

[13] NIST, National Vulnerability Database, https://nvd.nist.gov/.

[14] MITRE, Common Weakness Enumeration, https://cwe.mitre.org/.

[15] M. Schiffman, Common Vulnerability Scoring System, https://www.first.org/cvss.

[16] F. Baiardi, F. Tonelli, A. Bertolini, Cyvar: Extending Var-At-Risk to ICT, in: Proc. of the Int. Workshop on Risk Assessment and Risk-driven Testing (RISK 2015), Vol. 9488 of LNCS, Springer, 2015, pp. 49–62.

[17] F. Baiardi, F. Tonelli, A. D. R. Di Biase, Assessing and managing risk by simulating attack chains, in: PDP, IEEE Computer Society, 2016, pp. 581–584.

[18] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C. Wu., V. S. Tseng, SPMF: a Java open-source pattern mining library, Journal of Machine Learning Research 15 (2014) 3389–3393.

[19] F. Baiardi, J. Lipilini, F. Tonelli, Using s-rules to fire dynamic countermeasures, in: 2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), 2017, pp. 371–375.

[20] R. Srikant, R. Agrawal, Mining sequential patterns: Generalizations and performance improvements, in: EDBT, Vol. 1057 of Lecture Notes in Computer Science, Springer, 1996, pp. 3–17.

[21] F. Fumarola, P. F. Lanotte, M. Ceci, D. Malerba, CloFAST: closed sequential pattern mining using sparse and vertical id-lists, Knowl. Inf. Syst. 48 (2) (2016) 429–463.

[22] R. Agrawal, R. Srikant, Mining sequential patterns, in: ICDE, IEEE Computer Society, 1995, pp. 3–14.

[23] J. Chen, T. Cook, Mining contiguous sequential patterns from web logs, in: WWW, ACM, 2007, pp. 1177–1178.

[24] M. Norouzi, A. Souri, M. S. SamadZamini, A data mining classification approach for behavioral malware detection, Journal Comp. Netw. and Communic. 2016 (2016) 8069672:1–8069672:9.

[25] J. Wang, J. Han, C. Li, Frequent closed sequence mining without candidate maintenance, IEEE Trans. Knowl. Data Eng. 19 (8) (2007) 1042–1056.

[26] V. Grossi, A. Romei, S. Ruggieri, A case study in sequential pattern mining for it-operational risk, in: ECML/PKDD (1), Vol. 5211 of Lecture Notes in Computer Science, Springer, 2008, pp. 424–439.

[27] C. Luo, S. M. Chung, Parallel mining of maximal sequential patterns using multiple samples, The Journal of Supercomputing 59 (2) (2012) 852–881.

[28] P. Qin, L. Duan, T. Zhang, P. Wang, A parallel algorithm for mining density-aware distinguishing sequential patterns with Spark, in: CBD, IEEE Computer Society, 2016, pp. 144–149.

[29] F. Petitjean, T. Li, N. Tatti, G. I. Webb, Skopus: Mining top-k sequential patterns under leverage, Data Min. Knowl. Discov. 30 (5) (2016) 1086–1111.

[30] V. Grossi, A. Romei, F. Turini, Survey on using constraints in data mining, Data Min. Knowl. Discov. 31 (2) (2017) 424–464.

[31] M. van Leeuwen, J. Vreeken, Mining and using sets of patterns through compression, in: Frequent Pattern Mining, Springer, 2014, pp. 165–198.

[32] H. T. Lam, F. Mörchen, D. Fradkin, T. Calders, Mining compressing sequential patterns, Statistical Analysis and Data Mining 7 (1) (2014) 34–52.

[33] P. G. Srinivas, P. K. Reddy, A. V. Trinath, B. Sripada, R. U. Kiran, Mining coverage patterns from transactional databases, J. Intell. Inf. Syst. 45 (3) (2015) 423–439.

[34] Z. Zheng, W. Wei, C. Liu, W. Cao, L. Cao, M. Bhatia, An effective contrast sequential pattern mining approach to taxpayer behavior analysis, World Wide Web 19 (4) (2016) 633–651.

[35] C. Zhou, B. Cule, B. Goethals, Pattern based sequence classification, IEEE Trans. Knowl. Data Eng. 28 (5) (2016) 1285–1298.

[36] X. Liang, Y. Xiao, Game theory for network security, IEEE Communications Surveys and Tutorials 15 (1) (2013) 472–486.

[37] W. M. van der Aalst, Process Mining, Springer, 2011.

[38] W. M. P. van der Aalst, A. K. A. de Medeiros, Process mining and security: Detecting anomalous process executions and checking process conformance, Electr. Notes Theor. Comput. Sci. 121 (2005) 3–21.

[39] M. D'Andreagiovanni, F. Baiardi, J. Lipilini, S. Ruggieri, F. Tonelli, Sequential pattern mining for ICT risk assessment and prevention, in: SEFM Workshops, Vol. 10729 of Lecture Notes in Computer Science, Springer, 2017, pp. 25–39.

[40] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, U. Dayal, Multi-dimensional sequential pattern mining, in: CIKM, ACM, 2001, pp. 81–88.

[41] F. Esposito, N. D. Mauro, T. M. A. Basile, S. Ferilli, Multi-dimensional relational sequence mining, Fundam. Inform. 89 (1) (2008) 23–43.