Discovering and utilising expert knowledge from security event logs

Saad Khan[a,*], Simon Parkinson[a]

[a]*Department of Computer Science, School of Computing and Engineering, University of Huddersfield Queensgate, Huddersfield HD1 3DH, UK*

Abstract

Vulnerability assessment and security configuration of computer systems is heavily dependent on human experts, which are widely attributed as being in short supply. This can result in a system being left insecure because of the lack of easily accessible experience and specialist resources. While performing security tasks, human experts often revert to a system's event logs to establish security information (configuration changes, errors, etc.). However, finding and exploiting knowledge from event logs is a challenging and time-consuming task for non-experts. Hence there is a strong need to provide mechanisms to make the process easier for security experts, as well as providing tools for those with significantly less security expertise. In this paper, we present a novel technique to process security event logs of a system that have been evaluated and configured by a security expert, extract key domain knowledge indicative of human decision making, and automatically apply acquired knowledge to previously unseen systems by non-experts to propose security improvements.

The proposed solution utilises rule mining algorithms to extract security actions from event log entries. The set of identified rules is represented as a domain action model. The domain model and problem instance generated from a previously unseen system can then be used to produce a plan-of-action, which can be exploited by non-professionals to improve their system's security. Empirical analysis is subsequently performed on 21 event logs, where the acquired domain model and identified plans are discussed in terms of accuracy and performance.

Keywords: Event Logs, Association Rule Mining, Causality, Automated Planning

1. Introduction

Many organisations are facing security threats exposed by their digital infrastructure, and given the increasing size and critical nature of their business operations, there is a need to pro-actively identify and mitigate security vulnerabilities. This process requires expert knowledge of the latest security threats and how they can be mitigated. Such knowledge is in short supply, expensive, sometimes inaccessible and requires a high amount of human effort [1], but it is essential to keeping the system secure against malicious attacks.

The lack of expertise can potentially be resolved using computational intelligence to autonomously generate security evaluation and mitigation plans. Besides security, researchers from various other domains are exploring automation techniques, such as in mining specifications directly from software libraries [2], merging existing specification miners to produce a superior specification [3], inferring behaviour models from execution traces of object-oriented programs [4] and automatically discovering program flaws by learning method calls [5]. In this paper, an automated solution is presented that can extract security actions performed on a system using only event log entries. An event log entry contains information regarding the activities within the system, generated by either a user, application or the system itself [6]. Event logs are beneficial for tracking (step-by-step) state changes in the system (e.g. creating or deleting users), as well as troubleshooting the problems. This solution builds up the knowledge by discovering the chains of event relationships to represent specific human expert security actions. The relationships are termed as temporal-association-causal rules, which are achieved by incorporating a notion of time to frequently co-occurred

*Corresponding author

Email addresses: saad.khan@hud.ac.uk (Saad Khan), s.parkinson@hud.ac.uk (Simon Parkinson)

events and subsequently applying causal inference to rank discovered event relationships. The solution is currently tested on Microsoft Windows operating systems (OS) [7]; however, the approach is general, OS independent and can easily be applied to event logs of other systems. The data residing within Microsoft event log entries is structured and contains object-value pairs. A default set of classes exist in the .NET framework for identifying and parsing specific category of events, which facilitates an easier development and creation of testing environment for the proposed approach. Future extensions of this work will extract features focusing on other, widely used Linux- and Mac-based systems along with web-servers, OpenStack cloud [8] and Fog computing platforms [9].

Event logs are a powerful source of knowledge that can be used for understanding operations, performance and security [10]. According to Microsoft documentation[1], there are several categories of events in Windows-based systems: application, system, security and others. This paper only focuses on the security events, which are easily identifiable within the existing entries. The security events contain structured data that can be easily processed and provide detailed state information on a system's security controls. For example, failed login attempts, changes in user permissions, activity based on firewall rules, cryptographic operations and many other security-related events. The system activity is recorded based on the predefined security and configuration policies by the administrator. Each entry contains a series of objects that define a particular occurrence of an event. The objects include: a unique identifier (ID), event source, username, timestamp, machine name, and application and service-specific objects.

The security events provide knowledge that describes both unauthorised activities and configuration events. Many techniques have been developed in previous research to find relationships among events, and they generally utilise variations of clustering [11], correlation [12] and causal rule mining [13] algorithms. Most of the current solutions build a security model that can identify the causes behind system failures and implement remedial actions. These techniques are focussed on identifying problems with software solutions; however, there is a great potential to adopt this same philosophy for identifying security issues. It is likely that a security configuration against a certain issue generates a large number of events based on the specified audit policy, and produces a complex chain of events to record the activity. To the best of the authors' knowledge, there has been little research into automatically capturing and exploiting expert knowledge from this source.

It would be of great value to automatically identify chains of events against configuration activities. Consider a scenario where a system administrator is maintaining a server. A user illegally attempts to access a file named *abc.txt* and successfully views it. To document this activity, an event will be recorded with event type as 4663 depicting 'An attempt was made to access an object'. Assuming the administrator notices that the user is not supposed to access any file on this server and their permissions should be immediately removed. The administrator then modifies the network share permissions, and as a result an event log entry will be created with the event type as 4670, detailing 'Permissions on an object were changed'. The event logging mechanism generates this event in a way where it includes a record of the new and modified permissions. Many additional event log entries will also be generated in between 4663 and 4670, and therefore it would require expert knowledge, time and effort to realise the connection. Despite being simple, this example describes the fact that managing security requires experience and expert knowledge. A system that can automatically extract a relationship between 4663 and 4670 would be beneficial for the non-experts to directly look for such event entries and determine if any user is exercising more than the required set of permissions.

The research question addressed in this paper is whether it is possible to identify security-related actions performed by an expert and thus allow non-experts to utilise knowledge to audit and improve the security of their systems. In exploring this question, an unsupervised knowledge acquisition technique for identifying temporal-association-causal rules among event log entries is developed. It is a general approach that can be applied to any object-based event log dataset. It utilises correlation and causal mining algorithms due to their known suitability in finding such relationships [14, 15]. The solution first generates correlation rules, which are combined and ordered based on a temporal metric, and then used to produce cause and effect relationships. The final rules are encoded into a domain action model, which is subsequently used by an automated planning algorithm to determine a course of action on a previously unseen machine.

1.1. Contributions

The primary contributions of this paper are:

[1] Microsoft Windows documentation on Event Logging and Viewing: https://msdn.microsoft.com/en-us/library/bb726966.aspx

- **A novel technique to extract and represent knowledge from event log entries**. This technique creates an *object-based* model of event log entries, which are processed by an association rule mining algorithm. The rules are then converted into sequences of event relationships using a temporal metric, which are further validated by applying causal inference. The extracted knowledge is represented in the form of an action model and utilised against other, previously unseen computer.
- **A mechanism to automatically find support values for correlation mining in event log dataset**. The solution is capable of calculating minimum and maximum support values used during rule mining, derived from the object-based model of any event log dataset. The support value influences the amount and quality of correlation rules and it is subjective to the characteristics of a provided dataset.
- **Software implementation**. A novel software tool for mining Microsoft event logs, capable of learning knowledge from Microsoft event sources. The software represents the output of using acquired knowledge on previously unseen machines in a user-friendly manner.
- **Empirical analysis**. The proposed technique and tool is evaluated on 21 event log datasets from live, multi-user and network-based machines to determine overall performance and accuracy that is in the range of $73\% - 92\%$.

The document is organised as follows. Section 2 reviews and analyses existing domain learning techniques, automated planning, correlation mining and causal inference techniques, highlighting the importance of the proposed system. Section 3 details and motivates the pre-processing phase of the proposed system. Section 4 describes the process of association rule mining using an automatically calculated support value. Here association rules are represent by their respective event types, where any sequence of events are related through their objects. Section 5 presents our mechanism of filtering insignificant rules based on a temporal metric. It also shows the process of combining individual rules to form sequences of events, where each sequence represents a single configuration activity. Section 6 explains the process of forming a directed acyclic graph from the extracted rules, assigning a causal rank score to each rule based on the level or strength of causality. Section 7 presents a method to encode the extracted relationships into an action model, which leads on to the explanation of the automated method for extracting the problem instance from vulnerable machine, and applying automated planning technique to generate plan-of-action for security improvements. Finally, Section 8 performs empirical analysis on the event logs of 21 live systems and presents the performance and accuracy of the proposed solution.

2. Related Work

Many resources are readily available for individuals, who are seeking to gain knowledge and expertise in a security-related area. However, such type of knowledge sharing has the limitation that the user needs to know what skills they are lacking to conduct the security assessment and configuration of a particular system. Several solutions are available that extend and incorporate security knowledge from event log data sources to automatically identify and mitigate issues of a system. One such tool is Simple Event Correlator, which is based on a pre-defined knowledge-base of hard-coded rules to detect event relations [16]. The tool has various significant features for security management[2]. Another patented work built a multi-tier security event correlation and mitigation technique based on user-defined rules [17]. It identifies threats and complex attack patterns and presents effective security solution. Another research work implemented a static approach to determine the reasons behind certain system failures [18]. It applied rule-based classification and time-series methods on a specific set of event log entries to identify patterns. As the existing solutions mostly rely on manual rules, they will require expert knowledge and continuous management in terms of updating and constructing new rules.

2.1. Automated planning

Unlike conventional rule-based systems, the security knowledge can also be acquired and modelled using domain action learning [19]. Various tools are available to assist in the manual construction of domain models, for example,

[2]https://sourceforge.net/projects/simple-evcorr/

itSIMPLE [20] and GIPO [21]. Researchers have also produced autonomous knowledge acquisition tools, such as Opmaker [22] and Learning object-centric models or LOCM [23]. The learning process of these tools is dynamic and sometimes takes additional information, e.g. partial domain model and uncertainty factors, for building a complete domain model. Despite the advantages, a significant challenge faced in autonomous learning is from incomplete and noisy data sources. Further limitations also exist over the compatibility with domains having static facts, which remain unchanged throughout the planning process [24]. Recent advancements are starting to address the issues regarding static facts, such as in Automated Static Constraint Learner or ASCol [25]. The ASCol system exploits a directed graph representation of operators' arguments in a plan trace and uses that to discover pairwise relations among predicates.

The input used in the aforementioned systems is gathered from goal-based solutions and manual observations or experiments. The major drawback of state-of-the-art domain learning solutions is the need for pre-existing domain knowledge in the form of plan traces and human assistance. A plan trace is a the sequence of actions that represent a plan to achieve a desired goal. All existing solutions require plan traces, and as such performing autonomous learning in a previously unseen application area would first require the construction of a plan trace. In most domain engineering processes, this is not feasible because creating well-formed plan traces could be viewed as almost as challenging as constructing a domain model. More specifically, a plan trace can only be constructed one the domain is fully understood. In addition, it would also require expert knowledge, something which can not be guaranteed to be available. Techniques like crowd-sourcing models [26] and involving human experts [27] require large amount of effort to prevent human-error and perform conflict resolution. This motivates the research challenge of developing an autonomous approach to learn and utilise security domain knowledge directly from available system data sources, without prior knowledge and human aid. The output plan will be capable of helping non-experts and experts alike for the improvement of security in any new or previously unseen machines.

After acquiring the domain model, Automated Planning (AP) algorithms are employed to select and organise purposeful actions in achieving expected outcomes [28]. The AP algorithms are implemented in software solutions (names planners) and are used as general problem-solving techniques. They generate plans with respect to quality and time-based constraints [29]. The computer generated plans have shown more efficiency as they explore a wide array of possible solutions [30]. The Planning Domain Definition Language (PDDL) and its extensions are commonly used for encoding domain knowledge [31]. Recent studies show that AP is being used in numerous security related areas, for example, offensive attack planning in networks [32], model-based simulation of penetration testing [33], generating adversarial courses of actions [34] and threat mitigation plans [35].

2.2. Association rule mining

Considering the importance of creating an automated domain model, our research pursues potential data correlation techniques that are unsupervised and can discover relationships among a diverse set of event log entries. Several data mining techniques [36] are available that can serve this purpose, such as data clustering and rule mining. Data clustering techniques rely on item similarity metrics [37]. For numerical data, clustering algorithms employ methods like Euclidean distance, whereas for nominal data, set and character similarity methods are used. A previous study presents a novel clustering algorithm, called the Simple Logfile Clustering Tool [38], which identifies infrequent events based on outliers that could represent previously unknown fault conditions and other anomalies. The clustering is not a suitable technique for that of identifying knowledge in security event data sources as it will only create groups of similar items based on certain metrics for anomaly detection [39]. Furthermore, the groups do not necessarily represent security identification and mitigation activities. On the other hand, rule mining is a method of identifying patterns to reveal strong co-occurrences in the form of rules among seemingly unrelated items [40]. According to a study [41], there are 38 measures (e.g. support, confidence, lift, recall, specificity, etc.) that can be used either individually or combined to improve the quality of rule mining process. The proposed solution in this paper uses the Association Rule Mining (ARM) technique, which relies on support and other metrics to determine the correlations among data items. It enables searching for valuable information based on frequency and is combinatorial and symbolic in nature.

The first ARM algorithm [42] was developed to identify regular co-occurrences between the products in large-scale transaction data. The data was recorded by point-of-sale (POS) systems in supermarkets. The drawback is that it can only have one consequence statement in the rule i.e., it can generate $X \cap Y \rightarrow Z$, not $X \rightarrow Y \cap Z$ rules. Following this, two new breath-first search based algorithms were introduced called Apriori and AprioriTid [43]. Their main target was to improve the performance of ARM process in terms of execution time. Apriori was the first algorithm to pioneer support-based pruning and avoid the exponential growth of possible itemsets. The algorithm iterates over

sets of items in increasing arity; singletons, pairs, triplets etc. The general idea was, given an itemset $ABCD$, first extract their subsets ABC, AB etc. If a subset does not generate a rule, further subsets of ABC can be avoided based on the Monotonicity Principle [44]. Similarly, if any itemset is frequent, then every subset of the itemset will also be frequent (also know as the Apriori Principle). Apriori implements efficient pruning processes to prevent exponential growth of candidate frequent itemsets. Researchers have also a modified variant of ARM to identify irregular file system permissions [45]. This solution discovers the least frequent rules as oppose to finding strongly correlated rules like in most applications of ARM. This allows them to identify irregular or rather suspicious permissions that should be eliminated to improve data security.

It should be noted here that the presence of association rules among various mutually exclusive items of the dataset do not necessarily imply they are connected [46]. The rules are subjective to the given data (i.e., the results are local, not universal). A rule only depicts that there is a strong correlation between antecedent and consequent items. The quality of association rules can be improved by providing a meaningful mechanism to identify correlation, as evident by many studies. The temporality or time-based ordering can be used to identify or formulate interesting association rules among items [47]. It provides an additional piece of evidence for the relationship to exist. For example, finding whether event A always happens after the event B can help establish a higher confidence [48]. Different research studies have proposed tree-based algorithms that can perform temporal ARM [49]. The algorithms effectively reduce the computational cost by skipping the candidate item set generation phase. Another paper proposed a Multi-level Intensive Subset Learning (MIST) algorithm, which applies a temporal metric to correlation rules from financial time series [50]. The algorithm successfully identified profitable trades by generating association rules, which lead to uncover hidden knowledge. Another study presented a new algorithm, called T-Apriori [51], which includes a time-based constraint and is used for analysing ordered ecological events set. However, according to the paper, the efficiency of the algorithm has not been tested on larger datasets. It also requires an additional layer of data pre-processing using K-Means clustering algorithm, which might reduce the performance of overall solution. Another paper presents an algorithm, called ARMADA, which can discover time-dependent correlations or patterns within large datasets [52]. The authors claims that the temporal-association rules provide richer knowledge in terms of semantics, however, the algorithm was only tested on synthetic data.

A research study determined that the incorporation of temporal dependencies in rule mining process can group and (re)arrange rule items into correct semantic sequences [53]. Another study presents a conversion methodology from association to temporal-association rules given the spatio-temporal data [54]. It requires input of states and events of interest. The approach has been successfully tested on live systems where data belonged to fish movement in a river. A correlation technique for heterogeneous data has also been developed that groups the items based on properties and temporal precedence [55]. For testing, the data collected was in the form of events and belonged to various sources, such as Anti-malware tools, security scanner etc. Another paper [56] presents a solution for assisted living using temporal-association rule mining. The algorithm was used to recognise and build predictive activity models. Despite having significant advantages, the aforementioned approaches do not provide a suitable solution for identifying temporal-associations using a fully automated mechanism in raw data streams. They either rely on manual input or operate in a constrained and predictable environment to define connections. Some of the solutions require pre-configuration and pre-processing layers as well, which are not suited for large, complex models. Moreover, the requirement of manual input again leads towards needing the human expertise, which as discussed before are in short supply and lacks efficiency. The existing techniques are also incapable of defining the initial (starting) event in a sequence of multiple correlated events and hence do not provide the basis to develop an action model.

2.3. Causal relationships

The presence of correlation signifies a strong statistical relationship among the linked events, but it might not always be true. The relationship between events can be made certain by providing evidence of causality. The causal connection demonstrates that the events are conditionally dependent on each other and are manifested with the help of Markov chains [57]. A Markov chain is a sequence and description of random events in which the probability of each event depends only on the state reached in the previous event only. Various algorithms have been devised to find causality. The process of finding causal relationships can be classified into two broad categories: CCC and CCU triplets causality [58]. Assume there are variables A, B and C. The CCC mechanism is that all three pairs are correlated and one variable is known a priori to have no cause. For example, in case of (A, B), (B, C) and (A, C) pairs, if A has no causes then the causal relation will be A causes B and B causes C. The CCU mechanism is that two pairs are

correlated and one pair is uncorrelated. For example, if (A, B) and (A, C) are correlated and (B, C) are uncorrelated then B and C cause A. The CCU causality test is better in performance, while the CCC is better producing quality results. A study [59] presents Local causal discovery (LCD) algorithm, which is a constraint-based method to process observational data. It builds a Bayesian network by conducting independence tests and applies the CCC rule (but not the CCU rule). An improvement of LCD algorithm has been proposed in Bayesian Local Causal Discovery (BLCD) algorithm that reduces the amount of false positives [60].

Another paper used Bayesian method for learning the causality from observed and experimental data [61]. The proposed model successfully predicted the causal structure and parameters among randomly selected item-pairs by enforcing Causal Markov condition. Another study presents Relational causal discovery algorithm [62], which uses relational blocking instead of dependent or independent conditioning to determine causal connections. Blocking is a data grouping technique, which aims at reducing variation and determines common causes using the statistical measures. Another study presents Causal association rule discovery (CAR) algorithm [63], which applies retrospective cohort studies to association rule mining to discover persistent causal rules. There are also other causal mining techniques, for example, the Bradford Hill [64] and Granger [65] models. They propose that a causal rule should satisfy the following seven factors: association, consistency, temporal order, specificity, plausibility, coherence and have experimental evidence. Another approach to infer causality is presented by Popper that comprises of three conditions: temporal precedence, dependency, and no hidden variables. This approach [66] has been successfully exploited in the unbounded data streams of events to build causal networks.

3. Data pre-processing

This section presents the pre-processing phase of security event log entries. It has two steps and prepares the data for rule mining algorithms. The first step is to read security events log of a system and remove routine log entries. The second step is to build an object-based model from the remaining entries for further processing.

3.1. Filtering routine entries

During system use, a large quantity of the routine entries are created and stored, reporting on events that are not security configuration related. These entries do not contain object-value pairs and contain little, if any, information. For example, event 1105 is logged in the Microsoft system, when maximum log size is reached. This occurs when the operating system starts saving the new event log entries in another file, instead of over-writing the previous one. Although routine events provide valuable information for forensic analysis and system management, it can be the case that they do not depict any security expert knowledge or the reason behind why it was performed. In this work, such routine events are considered as *noise* due their relatively high frequency of occurrence and their strong relationship amongst a large volume of system users and resources. Furthermore, if these events are allowed to be processed in the rule mining process, their respective event entries will dominate and correlate with all other entries, hence having an negative impact on the techniques effectiveness.

The first step in this process is to create a frequency distribution (FD) of the event types in a given dataset. The size of FD will be equal to size of unique event types, where each element corresponds to the number of total entries of a particular event type. The following shows an example of FD from live system:

$$FD = \{152,35,158,2148,10,36,4,4,2121,32,8,44,44,23,1,1,1,8,2,1,2,1,26,1,2,2,5,152,1,1,1\} \quad (1)$$

Here we make an assumption: events that have occurred *significantly* more than others are routine. This assumption is based on empirical observation from several event log datasets. For a human being, it would be an easier task to determine that the event types against fourth (2148) and ninth (2121) elements of FD are routine, as their occurrence is relatively greater than others. For the similar automated detection and elimination of routine events, a Standard Deviation measure (s) can be used to determines how much the elements of a set differ from the mean value [67, 68]. A low s depicts that the elements are closer to the mean (μ), while a higher s value shows that the elements are dispersed over a wider range of values. The formula is shown in Equation 2:

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2} \quad (2)$$

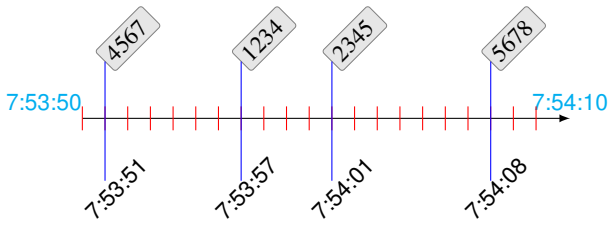

Figure 1: A visual time line representation of D covering a time span of 20 seconds.

where N denotes the number of elements, x_i is the i_{th} element and $\overline{x}$ is the average of FD. As the processed dataset is a segment of continuous stream of events, the proposed solution uses Sample Standard Deviation to form an estimate of larger population and output a generalised result. Hence, instead of N, which is used in the Population Standard Deviation, it uses $N-1$, which is known as Bessel's correction [69]. The ratio of the s to the μ is called the coefficient of variation (C_v), which is a measure of variability [70]. If the value of C_v is greater 1 (or 100%), it means the set contains one or more outliers [71], i.e., the frequency of certain event types is different than others. The routine event types can be filtered if all those event types are removed whose frequency is higher than s. For FD_1 in Equation 1, the s is 528.44, μ is 162.16 and C_v is 3.26. As $C_v > 1$, two elements (2148 and 2121) are found to be higher than s and hence the corresponding event types will be eliminated from the further processing. The remaining elements of FD_1 are shown in Equation 3.

$$FD_1 = \{152,35,158,10,36,4,4,32,8,44,44,23,1,1,1,8,2,1,2,1,26,1,2,2,5,152,1,1,1\} \tag{3}$$

3.2. Preparing object-based model

To enable a structured mechanism of processing event log entries, it is necessary to normalise their format and extract meaningful parts. In this work, an object-based model is adopted, where each event log entry is represented in terms of the system objects described in the event log entry. These could, for example, be user names, system resources, permission levels, etc. In the following discussion, D is used to model the dataset of event entries, where $D = \{E_1, E_2, ..., E_N\}$. The event, $E = \{id,O\}$, where id is a numeric event type, and O is the set of event objects. The set O belongs to set $I = \{o_1,o_2, ...,o_n\}$, such that $O \subseteq I$. Each entry, E_i, contains corresponding objects from I to represent an occurrence of event. Note that the event types are not part of D as it is only used for mining object-based correlation rules. For example, $D = \{E_1, E_2, E_3, E_4\}$, where:

$$E_1 = \{\{4567\},\{User1, Win7, Port : 53176, NTLM\}\}$$
$$E_2 = \{\{1234\},\{User1, ReadEA, svchost.exe, IKE\}\}$$
$$E_3 = \{\{2345\},\{User2, ReadEA, System, NTLM\}\}$$
$$E_4 = \{\{5678\},\{User2, Win7, NtLmSsp, Winlogon\}\}$$

In this trivial example, each entry has 4 objects along with their event types. It should be noted that the original event log entries are much more diverse and contain large number of objects. Events occur over the duration that a system is running and they appear in sequence, although concurrency of different processes can generate intricate entries. Each entry contains a time stamp to denote creation time. A sequential time line of the example is presented in Figure 1, where the events occur over a 20 second period.

4. Association rule mining

As discussed in the related work (Section 2), there is potential of using association rule mining (ARM) to identify relationships amongst the objects of event log entries. ARM is an unsupervised process [72], and employed as a method for describing, analysing and presenting association rules that satisfy the condition of (strong or rare)

co-occurrence. Identifying an association rule between the objects of two events will demonstrate the existence of connection between the events.

The application of ARM in the proposed solution consists of multiple steps. It starts by calculating minimum and maximum support values for ARM algorithm. Following on, object-based rules are first identified before being used to establish into event-based rules. More specifically, the strong relationships amongst event objects are utilised to discover correlation between events.

4.1. Object-based Association Rules

The purpose of object-based rule mining is to identify such objects that are likely to occur together. The rules are discovered in tabular dataset of objects that uses different measures to determine interesting data [73]. In the proposed solution, the Apriori algorithm is used for ARM as it finds complete frequent itemsets [74]. ARM takes two fundamental steps to extract usable ARs [75]:

- *Frequent itemset*: find the set of all items that appears more than a certain number of times, which is defined by the user, in data; and
- *Correlation*: identify co-occurrences among the frequent itemset, where the degree of strength is flexible and defined by the user.

Consider the set D of event log entries and the set I of total unique objects from Section 3.2. The fundamental idea behind generating correlation rules is to determine frequent co-occurrence of objects. They identify the different objects in event log entries that have appeared multiple times together. The following three steps are taken in order to discover interesting relationships [76]. In these steps, both X and Y contains objects from I. More specifically, $X = \{o_i, ...\}$ and $Y = \{o_j, ...\}$, where the values of i and j are between 0 and size of I.

Step 1 (Association rule). *$X \rightarrow Y$ is an Association Rule (AR), where X and Y contain one or more objects each from set I. X is the LHS (left-hand side) or body and Y is RHS (right-hand side) or head of rule. Both X and Y are disjoint sets of objects, i.e., $X \cap Y = \emptyset$. $X \rightarrow Y$ means that whenever an event entry contains X, then it probably contains Y too. In the continuing example, an AR would be $\{User1\} \rightarrow \{Win7\}$.*

Step 2 (Support of an AR). *An AR has support 's' in the event log dataset D, if s% of entries in D contain $X \cup Y$. Formally, it can be defined as: $Support(X \rightarrow Y)$, $s = \sigma(X \cup Y)/N$, where N is the total number of entries in D. The example AR has a support of $1/4 = 0.25$, since $\{User1\}$ and $\{Win7\}$ co-occur in 1 out of 4 event log entries.*

Step 3 (Confidence of an AR). *The confidence 'c' of an AR is defined as the ratio between the number of entries that contain $X \cup Y$ and the number of entries that contain X. Formally, it can be defined as: $Confidence(X \rightarrow Y)$, $c = \sigma(X \cup Y)/\sigma(X)$. The example AR has a confidence of $1/2 = 0.5$ as 1 entry contains both $\{User1\}$ and $\{Win7\}$, whilst, 2 entries contain $\{User1\}$. In other words, 50% of the entries that contains $User1$ are related to $Win7$.*

Researchers have demonstrated that the ARM generates a high quantity of rules and has a complexity of $O(N^2)$, where N is the total number of unique elements across all log entries [77]. It should be noted here that any rule with a required support count and confidence value is deemed relevant by the algorithm; however there might still not be interesting or useful to the users. The ARM process performs an exhaustive search for items that have strong co-occurrences in a given transactions. The term 'strong' is defined by the amount of support and confidence. According to an analysis of 61 interestingness measures on 110 different datasets [78], using a right support value with respect to a dataset is essential for producing high quality rules.

4.2. Establishing support value

Given the fact that ARM produces rules based on frequent items and has no prior knowledge about the dataset, it might not always extract meaningful rules. Events triggered due to the security activities may be lesser in number as compared to the routine events. This is because administrators do not often repeatedly perform the same security actions on a frequent basis. In addition, if support and confidence values are kept at minimal to identify low frequency activities, it would allow more routine events to appear in the generated rules. This motivates the need to define

both minimum and maximum support values, hereafter termed as support range (SR), at the same time to guide the algorithm to consider less frequent events whilst avoiding large number of routine events. As each dataset is different, we have devised an automated mechanism to determine the SR. The ARM algorithm is implemented in a way to take both minimum ($minsup$) and maximum support ($maxsup$) values. This ensures to find all those rules having $minsup \leqslant$ support $\geqslant maxsup$. Moreover, as the aim is to generate high quality association rules and the SR may come out as quite low (e.g. 3%-10%), the confidence is always set to maximum. The confidence value is an indication of how often the rule has been found to be true, and setting it to 100% ensures the discovery of the most interesting as well as reliable association rules [79] within the limits of SR.

We propose a new systematic mechanism of calculating SR using the object frequency distribution (OFD) of events types. Other techniques of a similar nature have also been developed [80, 81]; however, they do not calculate the threshold support value but rather rely on the higher confidence and lift values to mine interesting rules. Another disadvantage of existing techniques is the generation of large number of rules that have to be manually examined at later stages. The ARM process depends on the frequency of objects, which makes it necessary to use the frequency of objects, rather than frequency of events to estimate the SR. The OFD is calculated by determining the unique objects in a given event dataset, and then calculating the frequency of individual object's occurrence. The formula used to determine the SR from OFD is shown in Equation 4.

$$SR = \begin{cases} \frac{F_{min}}{F_{tot}} \quad to \quad \frac{F_{max}}{F_{tot}} & \text{if OFD is normal} \\ \\ \frac{F_{avg}}{F_{tot}} \quad to \quad \frac{F_{max}}{F_{tot}} & \text{if OFD is not normal} \end{cases} \tag{4}$$

where F_{min} is the minimum of OFD, F_{max} is the maximum of OFD, F_{avg} is the average of OFD and F_{tot} is the sum of all elements of OFD. In a normal distribution, the data points are in symmetrical order, and if the size of distribution is small, there will be a small difference between the minimum and maximum elements [82]. The minimum support value is calculated as the ratio of minimum frequency to the total of the OFD, while the maximum support value is calculated as the ratio of maximum frequency to the total of OFD. However, if the distribution is not normal, the minimum support value is calculated as the ratio of average frequency to the total of OFD, while the maximum support value is calculated as the ratio of maximum frequency to the total of OFD. If the same equation is used here to calculate the SR, the large difference between F_{min} and F_{max} will generate a wider SR, i.e., the F_{min}/F_{tot} value becomes significantly lower and that will subsequently force the ARM algorithm to include less interesting and redundant rules. Hence the distinction between a normal and abnormal distribution shown in Equation 4 is important. It will allow the ARM algorithm to include interesting and useful rules, meanwhile, preventing the extraction of irrelevant rules.

4.2.1. Normality test

Many methods are available to determine whether a given distribution is normal, such as Shapiro–Wilk (SW) and Two-Sample Kolmogorov–Smirnov (KS) tests. The distribution size can become large as there are hundreds of distinct events that can be triggered. The SW test only provides better results for small (50 or less) sample sizes [83]. Recent comparisons show that the KS test is an effective method among others [84, 85], and is suitable for large sample sizes. The KS test is used to decide if two underlying one-dimensional distributions differ from each other. The KS is a non-parametric test that quantifies a distance between the empirical distribution functions of samples [86]. The first step in the normality test is to produce a known standard normal distribution, which will be used as a reference against the object frequency distribution (OFD) of the dataset. We used an algorithm proposed by Kirkman [87] that either takes a pair of (standard deviation and mean) or (first and last elements) of the OFD to generate a reference normal distribution (RND). Hence the RND has similar values and range as of OFD. The next step is to compare the RND and OFD using Two-Sample KS test. It operates under the hypothesis that two samples come from a common distribution. If the hypothesis is accepted, then OFD is normal as RND is known to be normal. Otherwise if the hypothesis is invalid, then OFD is not normal.

The first step of two-sample KS test is to determine the empirical distribution function (EDF) [88] of both OFD

and RND. This uses the formula provided in Equation 5.

$$EDF_n(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{x_i \leq t} \tag{5}$$

where $\mathbf{1}_{x_i \leq t}$ is the indicator function. It is a step function and outputs 1 if $x_i \leq t$ is true or else 0. For example, consider the OFD shown in Equation 6.

$$OFD_1 = \{1,2,3,4,14,22,31,59,60,68,91,144\} \tag{6}$$

The total number of objects in OFD_1 is 12, minimum value is 1, maximum value is 144, average is 41.58 and sum of all elements is 499. The empirical distribution, EDF_{OFD_1}, is presented in Equation 7.

$$EDF_{OFD_1} = \{0.08,0.17,0.25,0.33,0.42,0.5,0.58,0.67,0.75,0.83,0.92,1.00\} \tag{7}$$

In this manner, the EDF is applied to both OFD and RND to discover EDF_{OFD} and EDF_{RND} values, respectively. The next step is to find the maximum set of distances (or differences) between the items of EDF_{OFD} and EDF_{RND} using Equation 8:

$$D = \sup |EDF_{OFD} - EDF_{RND}| \tag{8}$$

where *sup* is the supremum or maximum of found distances. The maximum distance value, D, is used to determine if the hypothesis is acceptable or void by calculating a critical value using the Equation 9.

$$D > 1.36\sqrt{\frac{n+m}{nm}} \quad \text{(critical value)} \tag{9}$$

where n is the size of EDF_{OFD} and m is the size EDF_{RND}. If D is greater than the critical value, the hypothesis is acceptable, and hence the OFD will be considered as a normal distribution, otherwise not normal [89]. This process is repeated with multiple, distinct reference normal distributions that are generated from the same pair of values from OFD. The dominant outcome of hypothesis, which is either true or false, is considered as a final outcome. The multiple tests improves the consistency of Two-Sample KS tests, which consequently provisions better support values for correlation mining. Based on the normality test, i.e., whether OFD is normal or not, the Equation 4 is used to define a range of support values. In case of OFD_1, the process is repeated 12 times (as there are 12 elements) and the dominant output comes out as false. This means OFD_1 failed the Two-Sample KS test and OFD_1 is not considered as normal. Using Equation 4, the minimum support value would be $\frac{41.58}{499} = 0.08$, whereas the maximum support would be $\frac{144}{499} = 0.29$. The minimum and maximum support values will be calculated for every given event log dataset in this manner and given to ARM algorithm to mine object-based association rules.

4.3. Event-Type Association Rules

At this stage, correlations amongst event objects have been identified, and the next stage is to translate these relationships to identify connections among events. The object-based rules describe how objects contained in event entries are related in a particular machine; however, the aim of this work is to determine generic relationships amongst event types.

As previously defined and repeated to increase readability, object-based rules are in the form of $X_i \rightarrow Y_i$, where $(X_i, Y_i) \subseteq O$ and $X_i \neq Y_i$. The process of converting them into event-type rules requires matching objects belonging to both X_i and Y_i separately within all event entries, and extracting the event types of matched entries. This matching process accounts for both similarities and dissimilarities by considering: the number of matched objects (*match*), the number of objects missing from a rule but exists in the event entry (*missing*) and number of objects that are in the rule but missing from event entry (*additional*). The formula used to determine similarity is $(matched/(matched + missing + additional))$. If a similarity of 0.70 or above is found between the objects of X_i or Y_i of a rule and event entry, it is considered a strong match to extract feasible event types. The similarity threshold is

flexible and depends on the user, where a high value will generate higher quality rules, where as a lower value will generate a higher number of rules that might be of a lower quality. For example, assume $X_0 = \{a\}$, $X_1 = \{a, b, c, d, e, g\}$ and an entry $E_0 = \{a, b, c, d, e, f\}$. All elements of X_0 are fully matched with E_0, even though X_0 is missing 5 elements that are present in E_0. Hence the similarity amount will be $\frac{1}{(1+5)} = 0.167$. On the other hand, the elements of X_1 are not a full match with E_0, but the similarity amount is $\frac{5}{(5+1+1)} = 0.71$. Therefore the rule with X_1 will be replaced by E_0. In this manner, the proposed solution processes LHS and RHS of all object-based rules, which are then replaced with corresponding event types. There is a possibility of same event type occurring in both LHS and RHS. In that case, the one with the highest similarity is kept the other one will be removed.

The correlation rules among events are now grouped together. It is clear that relationships exist between events as evident from the object-based rules. However, the information of temporal ordering of events, i.e, which event occurred first, second and so on, is missing from the rules. This is problematic as temporal ordering is important to identify the order by which events occurred, and thus the order of actions performed on the monitored system. For now, due to unknown temporal ordering, an undirected edge symbol $(-)$ is used to represent the undefined direction in the event-based association rules. For example, in an object-based rule, if the LHS objects match with 1234 and 4567 event types, while the RHS objects 5678, the resultant event-based rule will be denoted as $\{1234, 4567\} - \{5678\}$.

5. Sequences of event relationships

This section presents the process of identifying the order of events in each rule based on a temporal metric. Following on, individual rules are formulated into sequences of events which are later validated.

5.1. Temporal-Association Relationships

Now that the event-based correlation rules have been established, it is necessary to determine the ordering of events within each association relationship. The event logging mechanism attaches a timestamp with every event log entry. The proposed algorithm utilises a temporal metric for ordering, which is based on the timestamp of event log entries. In other words, the ordering of events in all rules is determined based on when the events were generated. Similar approaches have been successfully applied to improve the quality and efficiency of various pattern discovery algorithms in sequential data streams [90].

Algorithm 1 presents the implemented technique which takes the event-based rules and the event log dataset as input and outputs temporal accuracy values of the rules. It starts by iterating over event-based rules on line 3 and determines all pairwise subset combinations between the items of LHS and RHS on line 5. For a rule $(x_1, x_2) - (y_1, y_2)$, the subset would be $(x_1 - y_1)$, $(x_1 - y_2)$, $(x_2 - y_1)$ and $(x_2 - y_2)$. The total number of subsets is the product of the number of elements on LHS and RHS, which is $2 \times 2 = 4$ in the example. The main reason for finding the subsets is to determine if there exists a temporal link between each pair of correlated events. Notice that processing each subset combination of all rules is computationally expensive. However, it is a trade-off between the quality of rules and the time and effort required to generate them. Processing each subset combination will reduce the risk of missing any interesting rule. Upon finding the appropriate ordering, the subset combinations of the itemset in the rule would be considered for further processing, or else it would be discarded.

The next step is to determine the temporal-association accuracy (TAA) of all subsets from each event-based rule. This will facilitate the conversion of correlation rules into temporal-association connections. The TAA depict the number of times a certain relationship was found accurate with respect to the dataset, i.e., correct event ordering based on the temporal sequencing of events. Pearl [91] also demonstrated that the temporal ordering of entities can provide beneficial results in inferring the practical and useful event connections. Processing events based on their temporal sequence does introduce a degree of uncertainly as we cannot measure reliability of the temporal sequence. For example, it could be possible that the event logging processes operate at a low system priority and the process of raising an event may be queued during high priority processing of other applications.

For every subset, the indices of event type from LHS (line 6), as well as RHS (line 7) are determined and saved in $PosE_x$ and $PosE_y$ lists, respectively. The indices are acquired from database D and sorted in timely order. After that, by comparing the elements of $PosE_x$ and $PosE_y$ lists, calculate the number of times each LHS event occurred before every RHS event and save the count in t_f, as shown in line 8. Similarly, determine how many times each RHS

Algorithm 1 Identifying and filtering temporal-association rules.

Input: Set of event-based rules $R = \{r_1, ..., r_n\}$, where $r = (r_x - r_y)$, and r_x and r_y consist of at least one $EventType$ each

Input: Set of ordered event log entries D containing 2-tuple of $EventTypes$ and their corresponding objects, which were used in creating object-based associative rules

Output: Set of temporal-association rules $C = \{(c_1, TAA_1), ..., (c_n, TAA_n)\}$, where $c = (c_x \rightarrow c_y)$ and c_x results in c_y event and TAA is the temporal accuracy of relationship

1: **procedure** TEMPORAL-ASSOCIATION-RELATIONSHIP
2: Initialise $C \leftarrow \emptyset$
3: **for all** $r_i \in R$ **do**
4: $(r_x, r_y) \leftarrow r_i$
5: **for all** $EventType_x, EventType_y \in r_x, r_y$ **do**
6: $PosE_x \leftarrow$ **GetIndicies**$(EventType_x, D)$
7: $PosE_y \leftarrow$ **GetIndicies**$(EventType_y, D)$
8: $t_f \leftarrow$ **Count**$(\forall x \in PosE_x < (\forall y \in PosE_y))$
9: $t_s \leftarrow$ **Count**$(\forall y \in PosE_y < (\forall x \in PosE_x))$
10: Initialise $TAA \leftarrow 0$
11: Initialise $direction \leftarrow 0$
12: **if** $t_f > t_s$ **then**
13: $TAA \leftarrow t_f/(t_f + t_s) \times 100$
14: $direction \leftarrow 1$ ▷ means $X \rightarrow Y$
15: **else if** $t_f < t_s$ **then**
16: $TAA \leftarrow t_s/(t_f + t_s) \times 100$
17: $direction \leftarrow -1$ ▷ means $Y \rightarrow X$
18: **end if**
19: **if** $TAA \geq 50$ **and** $direction$ is 1 **then**
20: $C.Add((EventType_x, EventType_y), TAA)$
21: **end if**
22: **if** $TAA \geq 50$ **and** $direction$ is -1 **then**
23: $C.Add((EventType_y, EventType_x), TAA)$
24: **end if**
25: **end for**
26: **end for**
27: **end procedure**

event occurred before every LHS event and count the value in t_s (line 9). If the value of t_f is found to be greater than the value of t_s on line 12, it means the LHS event (mostly) occurred before the RHS and the direction of rule will be $X \rightarrow Y$. Otherwise the direction will be $Y \rightarrow X$ due to else condition on line 15. In case t_f and t_s are equal, the correlation rule becomes ambiguous and the subset rule is ignored. The reason behind this two-way comparison is to find the direction based on temporal validity and individually establish the TAA of every subset. The TAA value is calculated as a percentage of times any correlation rule was found correct as shown in lines 13 and 16. Every output subset rule has 50% or more above TAA value. All subset rules are accumulated in a set C for further processing along with the newly found directions (lines 20 and 23) between event relationships.

The 50% TAA threshold value means that only those temporal-association rules will be selected, where half of the times or more, LHS event occurred before RHS or vice-versa. The purpose of the TAA value is to show the accuracy of respective relationship, regardless of what it represents. From a deterministic view point, any rule whose TAA value is less than 100% might be spurious, but 50% threshold value was chosen for three major reasons. First, higher threshold values lead to empty results in some datasets, which may be due to the large amount of noise consisting of routine, repetitive log entries. This produces a relatively inconsistent object-model and therefore makes it difficult to obtain rules with 100% TAA. Second, rules with a lower TAA value can still provide beneficial knowledge to user. Third,

choosing TAA value above 50% means that the LHS event occurred at-least once before in time than the RHS event or vice-versa in other case. Hence we selected this threshold value create a balance between the quality and quantity of rules by tolerating somewhat 'inaccurate' rules, rather than having none at all. The threshold value is flexible and can range from 50%-100%. This can either increase or decrease the quality results; however, 50% is sufficient based on our empirical analysis.

For example, consider a hypothetical dataset containing 200 event log entries, and the solution outputs (1234,4567−5678) event-based rule (as explained in Section 4.3). The first step is to divide the rule into two subsets (1234 – 5678) and (4567 − 5678). Considering the rule subset (1234 − 5678), the event type from LHS (1234) might potentially occur at $E_9, E_{36}, E_{59}, E_{73}, E_{105}$ and the event type from RHS (5678) at $E_{21}, E_{43}, E_{57}, E_{88}, E_{112}$. The t_f and t_s values of (1234 − 5678) would be 14 and 11 respectively. As the t_f value is greater than t_s, the final temporal-association would be (1234 → 5678) with $14/(14 + 11) \times 100 = 56\%$ TAA. Similarly, assume the same process is repeated on (4567 − 5678) and the other subset comes out as 4567 → 5678 with TAA value of 70%.

5.2. Forming and Validating Sequences of Events

Now that we have a set of temporal association rules, the next step is to consider their relationships as it is probable that the underlying security task performed by a human expert will be described by more than two events. In this section, we present a process to devise sequences of temporal association relationships. The temporal-association sequences present a complete set of events that were triggered while conducting security-related activities. The approach starts by iterating over all subsets $\{(c_{x_0} \rightarrow c_{y_0}), \ldots, (c_{x_n} \rightarrow c_{y_n})\} \subseteq C$ and creates clusters of subsets that have common RHS event type. After that, the solution combines the LHS event types of each cluster. For example, consider the following subset rules: $(c_{x_0} \rightarrow c_{y_3})$, $(c_{x_1} \rightarrow c_{y_3})$ and $(c_{x_2} \rightarrow c_{y_3})$. The combined subset rules are: $(c_{x_0}, c_{x_1}, c_{x_2} \rightarrow c_{y_3})$. This outputs a group, G, containing the combined subsets. Each member $g \in G$ will have one or more event types on LHS linking a single event type on RHS. The purpose is to connect all those events together, which lead towards the same goal. Although at this point, the combined events represent the unordered steps to perform one or more particular actions. Continuing the same example from Section 5.1, $g = (1234, 4567 \rightarrow 5678)$, and G would be equal to g as there are only two subsets in total. Set G implies that all events from LHS $(1234, 4567)$ lead towards a single event on RHS (5678).

The next step is to create an ordered set of events within every $g \in G$, so that the sequence of correlated events can be formulated into a chain. The first step is to find those two event entries having a maximum time difference. It is assumed that those two events will mark the starting and ending events of action(s) that were performed on the underlying system. Similarly, identify the second to last event based on the time that had happened before the ending event. Repeat the process until the last event is reached. This process will arrange the event entries within g in terms of time, therefore defining the initial set of temporal-associations. Referring back to Figure 1 for the ongoing example, it can be seen that the time difference between 1234 and 5678 is 11 seconds and 4567 and 5678 is 17 seconds. This means 4567 occurred 6 seconds before 1234, which is why it would be considered a starting point. The final chain or sequence of temporally associated events would be (4567 → 1234 → 5678).

The next step is to validate the extracted sequences of events. This process gathers and determines a TAA value for each pair of the sequence. After that, it takes an average of all TAA values, which is considered an overall TAA for the sequence. Continuing the example, TAA values will be calculated between 4567 → 1234 and 1234 → 5678. Assume the values are 70% and 80% respectively, the overall TAA of the sequence will be $(70 + 80)/2 = 75\%$. Hence the chains of events are formulated using frequent pattern observation of events (ARM) and temporal evidence yielded by Algorithm 1.

At this stage, temporally associated events have been arranged into appropriate sequences, which can describe the set of actions required to perform a certain assessment or configuration task. However, according to the Simpson Paradox [92], correlation does not necessarily imply causality. The presence of correlation between events does not suggest that one event is the result of the occurrence of the other event. If two events, say 4567 and 1234, are not in an associative relationship, they cannot form a cause and effect relationship. Also, if both events are associated, but the correlation is falsified when a third event 5678 is introduced as they are still not causally connected. It means that 4567 and 1234 are dependent on 5678 and 4567 is not a direct cause of 1234. Hence it is necessary to identify the most feasible cause, amongst a group of different correlated events, as multiple events can lead to a single event. The discovery of cause and effect relationships will indicate the progression of events, such that the first event is (fully or partially) responsible for the second event. Moreover, the additional information of whether a certain correlation is in

cause and effect relationship will increase the confidence in extracted knowledge. This motivates the need to find the strength of causality in the extracted sequences of events.

6. Causality

Causality may have different meanings for different datasets, which makes it difficult to discover causal connections in a unified and standard form. The causal relationships are also hard to explain as their justification depends on intuition. In the presented work, causality refers to the relationship of two or more events, such that they inform of a finite sequence of actions, which were performed in the same order, to improve the security of underlying machine.

Many techniques have been introduced in previous years that can discover causal rules. Most of the algorithms belong to either Bayesian Networks or Constraint-based approaches. In a large and complex observational data, Bayesian solutions require high computation power due to the presence of latent (hidden) variables [93]. However, constraint-based approaches have been widely utilised in several real-world problems as they are generalised and have lower execution time [94]. In our proposed solution, we first create a directed acyclic graph from the given set of associative relationships as it is the requirement of causal inference algorithm. This process includes the careful elimination of associations, which do not satisfy the criteria. Following on, an algorithm is used to determine the causality of each relationship. The rules with higher causality values are considered more reliable.

6.1. Building Directed Acyclic Graph

A graph (G) is defined as a set of vertices (V) and edges (E), i.e., $G = (V, E)$, where $V = \{1, \ldots, n\}$ and $E \subseteq V \times V$. The set of edges is a subset of all ordered pairs of distinct nodes. In our solution, N corresponds to all unique event types and E represents their associations. An edge between event type i and j is called directed if $(i \rightarrow j) \in E$ but $(j \rightarrow i) \notin E$. A directed acyclic graph (DAG) is a graph G where every edge is directed, does not have conflicting edges and cycles. In the proposed solution, it is important to create a DAG as the conflicts and cycles would present inaccurate, confusing and recurring event relationships, instead of providing useful knowledge. The remaining section explains the procedure of creating DAG.

The first step is to iterate over all temporally associated sequences and create subset (one-to-one event) rules of relationships. This atomisation process makes it easier with respect to implementation for further processing. The next step is to remove conflicting subset rules based on temporal-association accuracy (TAA) value. For example, if $E_1 = (i \rightarrow j)$ and $E_2 = (j \rightarrow i)$, and both edges exist in G, determine the TAA for each edge. If $TAA_{E1} \geq TAA_{E2}$, then remove E_2, otherwise E_1.

The next step is to remove cycles from the graph G. A cycle is defined as the path of edges and vertices, wherein at least one vertex is reachable from itself, and the vertices and edges cannot repeat. The proposed solution performs topological sorting using Kahn's Algorithm to detect cycles [95]. Kahn's algorithm first determines the number of incoming (in-degree) and outgoing (out-degree) edges for each vertex. It also maintains an ordered list of vertices, whose in-degree is or becomes zero later on in the process. It starts by traversing the neighbours of a vertex, whose in-degree is initially zero. For every traversed neighbour of the vertex, the out-degree of the vertex and in-degree of the neighbour is reduced. The vertices are added into the list as soon as their in-degree becomes zero. In case there is a loop in the graph, the in-degree of any vertex on a cycle never becomes zero, and it would not be added into the sorted. In current settings, we take all such (remaining) edges and rank them on basis of TAA value. An edge with a lowest TAA is eliminated, and the graph is checked again for cycles. This process is repeated until the graph becomes completely acyclic.

Consider the event temporal-association rules and their TAA values (in brackets) presented in Table 1. The events are represented in terms of A, B, C and so on. There are nine rules in total and the table informs the three-stage process of converting those rules into a DAG. In the first stage, conflicting rules are removed based on the TAA value. The rules $(A \rightarrow B)$ and $(C \rightarrow B)$ are in conflict with $(B \rightarrow A)$ and $(B \rightarrow C)$ respectively, where the later two rules have lower TAA values and hence eliminated from further processing. In the second stage, the rule $(B \rightarrow E)$ occurred two times (at number 4 and 8), and one of them is removed. In final stage, Kahn's algorithm discovered a cycle containing the rules $(B \rightarrow E)$, $(E \rightarrow D)$ and $(D \rightarrow B)$, where TAA values are 1, 0.5 and 0.8 respectively. Within these rules, the one with the lowest TAA will be removed, i.e., $(E \rightarrow D)$. The remaining rules are shown in the last column of Table 1, which represents the formation of (input) temporal-association rules into a DAG.

Original rules	(Stage-1) Remove Conflicts	(Stage-2) Remove Duplicates	(Stage-3) Remove Cycles
1. $A \rightarrow B$ (1.0)	$A \rightarrow B$	$A \rightarrow B$	$A \rightarrow B$
2. $C \rightarrow B$ (0.9)	$C \rightarrow B$	$C \rightarrow B$	$C \rightarrow B$
3. $D \rightarrow B$ (0.8)	$D \rightarrow B$	$D \rightarrow B$	$D \rightarrow B$
4. $B \rightarrow E$ (1)	$B \rightarrow E$	$B \rightarrow E$	$B \rightarrow E$
5. $F \rightarrow E$ (0.7)	$F \rightarrow E$	$F \rightarrow E$	$F \rightarrow E$
6. $B \rightarrow A$ (0.5)	$\times$	$\times$	$\times$
7. $B \rightarrow C$ (0.6)	$\times$	$\times$	$\times$
8. $B \rightarrow E$ (1)	$B \rightarrow E$	$\times$	$\times$
9. $E \rightarrow D$ (0.5)	$E \rightarrow D$	$E \rightarrow D$	$\times$

Table 1: Example of converting temporal-association rules to a DAG by removing conflicts, duplicates and cycles.

6.2. Inferring Causal Rank

There are several algorithms available to perform causality analysis in a graph, such as the Peter-Clark algorithm (PC) [96] and the subsequent improved version named Fast Causal Inference (FCI) [97]. In our work, the FCI algorithm is chosen, which allows hidden variables and has good scalability with high-dimensional data. The first part of FCI is same as the PC algorithm, so the following section starts by explaining the PC algorithm and then proceeds on to the FCI algorithm. Both PC and FCI are the commonly used constraint-based algorithms [98] and search for causal constraints among all vertices and edges of a DAG and present a unified model of causality.

6.2.1. PC algorithm

The PC algorithm has been used in several real-time applications, such as analysing orthopaedic implant data [99]. An efficient implementation of the PC algorithm is presented by Kalisch and Buhlmann [100]. It consists of the following steps:

1. Create DAG skeleton;
2. Perform conditional independence tests;
3. Orient v-structures; and
4. Orient other remaining edges.

The skeleton of a DAG is created by converting all directed edges into undirected ones of a graph. Consider three events X, Y and Z, where X is connected to Y and Y is connected to Z. Regardless of the direction of relationships in this graph, the PC algorithm considers them as $X - Y - Z$ (skeleton). Processing the skeleton of a DAG creates an additional layer of validation and assertion (i.e. a new criterion) of previously acquired event relationships as the directions are nullified for this stage. This allows the inference of causal rank without any bias or influence from the Temporal-association-accuracy (TAA). Following on, the algorithm creates an adjacency matrix of the skeleton and performs a conditional independence (CI) test between each adjacent pair of vertices (events) in the skeleton. An adjacency matrix is a $N \times N$ matrix that is used to represent a finite graph of N vertices. The elements of the matrix depict whether pairs of vertices are connected in the graph. The CI test based on the Causal Markov Condition, i.e., given parents, the event is conditionally independent of all non-descendants or the event is conditional on its parents (its direct causes), the event is independent of every other event, except its effects.

The PC algorithm employs d-separation technique to perform CI test. The 'd' in d-separation stands for dependence. It claims that two variables X and Z are d-separated relative to a set of variables Y in a directed graph, then they are conditionally independent on Y in all probability distributions such a graph can represent [101, 102]. In other words, the events X and Z are d-separated if all paths that join them are *inactive* relative to other vertices, or simply, there is no *active* path between them. This means that the knowledge of whether X occurs provides no information of Z occurring and vice versa. In our settings, the CI property means that the related events in a sub-path of graph are not related to each other, and hence either the whole sequence of events is incorrect or it is missing some crucial events that

1. $X \rightarrow Y \leftarrow Z$

2. $X \rightarrow Y \rightarrow Z$

3. $X \leftarrow Y \leftarrow Z$

4. $X \leftarrow Y \rightarrow Z$

Figure 2: Possible edges between a pair of variables (X and Z) given a third variable (Y)

Table 2: Determining the active and inactive paths

	Collider	**Non-Collider**
Empty Conditioning set	Inactive	Active
Non-Empty Conditioning set	Active	Inactive

are necessary to define events sequence (i.e. an expert's security action). The relationships where the Causal Markov Condition is satisfied are the least reliable with respect to causality. In the original PC algorithm, the conditionally independent vertices and respective edges are eliminated from the graph. However in the proposed solution, we keep track of such relationships in another set and assign them a low priority. These relationships, although having a low priority, are still temporally associated and might bring interesting information to the user.

After the CI tests on adjacency matrix, the PC algorithm applies orientation rules to the skeleton. The orientation is a process of assigning direction to the undirected edges, forming an equivalence class of the DAG [103]. The first step is to consider each triple of vertices X, Y and Z, such that the pair X, Z and Y, Z are each adjacent in the skeleton but X, Z are not, based on Y being empty or not. Consider the first path presented in Figure 2 in terms of X is conditionally independent of Z, i.e., $X \perp Z$. Both events X and Z cause Y, but there is no connection between them. This is known as inactive path or conditionally independent, where Y is called collider in the theory of d-separation [104]. This path will be assigned a low priority by the proposed solution. Now consider the same path again, but assume that the event Y is already known or observed. It means that Y has now become a condition for X and Z ($X \perp Z|Y$), so the conditioning set, unlike before, is not empty. The independent causes (between X and Y and Z and Y) are made dependent by conditioning on a common effect. Knowing $X \rightarrow Y$ can provide information about event Z, as $Y \leftarrow Z$ is the only other connection left, which makes this path active. Hence such triplets of vertices will be oriented into $X \rightarrow Y \leftarrow Z$ shape. This orientation is called 'v-structure'.

After identifying all v-structures, the PC algorithm directs the remaining edges, since it can deduce that one of the two possible directions of the edge is invalid because it introduces a new v-structure or a directed cycle. The directions are assigned based on whether a path is active or inactive between a pair of items from adjacency matrix. Consider the remaining paths (2, 3 and 4) of Figure 2 with respect to empty conditioning set, i.e., Y is unobserved. For the second path, the event X is an indirect cause of event Z. For the third path, Z is an indirect cause of X. For the fourth path, Y causes both X and Y. All of these paths are active, as Y is the non-collider. However, if the conditioning set is non-empty, i.e., Y is known, event X can only cause Z, if and only if event Y occurs in the second path. Regarding the third path, Z can only cause X, if Y occurs. Similarly in the fourth path, only Y can trigger X and Z. In a way, Y is blocking a connection between X and Z in all of these paths, hence rendering them inactive. All these paths will be assigned a low priority due to d-separation. Table 2 provides a summary of finding active and inactive paths based on whether the conditioning set is empty or non-empty.

Based on the above discussion, the PC algorithm assigns directions to the skeleton of the DAG. However, the presence of hidden variables limits the algorithm to consider only subsets of the adjacent nodes of X and Z to decide whether they are d-separated. It is therefore safe to assume that the initial skeleton may contain some superfluous and unprocessed edges due to incomplete CI tests that should also be eliminated (or assigned low priority in our case). Such edges can be determined by the FCI algorithm, which is detailed in the following section.

6.2.2. FCI algorithm

The FCI algorithm, which is an extension and generalisation of PC algorithm, takes the output of PC algorithm and performs additional CI tests [105]. The PC algorithm alone might produce incomplete results due to the presence of hidden variables in the DAG that are responsible for causality. A hidden variable, sometimes also referred as confounding or latent, is an unrelated or rather meaningless variable that correlates (directly or inversely) with both dependent and independent variables, i.e., it is an unobserved hidden common cause [106]. Without exposing all hidden variables, one cannot infer a causal connection among variables as there might be other factors influencing the relationship. The FCI algorithm includes additional orientation rules to discover and represent such relationships, and consequently generates complete and sound output [107].

To perform the additional CI tests between the items of an edge (e.g. X and Z), the FCI algorithm applies two functions; Possible-D-SEP(X, Z) and Possible-D-SEP(Z, X). The Possible-D-SEP (PDS) of X and Z is defined as follows: any vertex V_i of graph G will be in PDS, if there is a path ϕ between V_i and X or V_i and Z, such that ϕ contains a collider or cycle in one of the sub-paths. Hence all of those vertices that are not yet determined to be conditionally independent are placed in PDS. The CI tests are performed for arbitrarily many latent variables that are found in PDS. This larger exploration leads towards discovering relationships that were missed by the PC algorithm. Secondly, any vertex that is not PDS does not require the CI test [108]. Hence the FCI algorithm is capable of providing complete and sound set of causal relationships in a reasonable amount of time.

After the (additional) CI tests, the FCI algorithm uses the Maximal Ancestral Graphs (MAGs) mixed graph technique for representing the collection of all causal relationships. A mixed graph contains all directed, bi-directed and undirected edges. An ancestral graph is a mixed graph, where there is no vertex X, which is an ancestor of any of its parents nor any of its spouses. The MAG is a type of ancestral graph, in which for every pair of adjacent vertices X and Z, there exists one or more vertices Y that 'm-separates' them [109]. The m-separation is a generalisation of d-separation and it is equivalent of d-separation in DAG. It measures the graphical disconnectedness in MAGs. To define m-separation, first consider the definition of its opposite 'm-connecting'. A path p between two distinct X and Z vertices of the ancestral graph is m-connecting relative to Y, where $(X, Y) \notin Y$, if (a) every non-collider on p is not a member of Y and (b) every collider on p has a descendant in Y. The two vertices X and Z will be m-separated given Y in G, if there is no path m-connecting X and Z. In a nutshell, the FCI algorithm uses m-separation to generate a set of equivalent Maximal Ancestral Graphs (MAGs) from the input (DAG) to encode all conditionally independent relations found by (1) d-separation and (2) Possible-D-SEP with respect to latent variables [110].

The MAGs represent marginal independence (MI) models of DAGs. The MI means that only two variables (X and Z) are considered for independence while the third (Y) is completely ignored, i.e., the knowledge of event Y does not affect the occurring of events X and Z. Based on MI, causally different graphs can result in the same set of conditionally independent relations. These graphs are called to be Markov equivalent [111]. The multiple MAGs generated by the FCI algorithm have same adjacent vertices and (usually) common edge orientations. A single MAG does not present all causal connections, and hence it is not fully testable with observational data. This raises the need for a unified graph that is capable of representing all MAGs. The unified graph is called Partial Ancestral Graph (PAG), which is Markov equivalent to all MAGs [112].

A PAG can have three end marks for edges: arrowhead (>), tail (−) and circle (o). It can form four kinds of edges: $\rightarrow$, o−o, $o\rightarrow$ and $\leftrightarrow$. The PAG has the same set of adjacencies as of MAGs and represents different types of causal connections found by FCI algorithm. The presence of any edge shows the conditionally dependent relationship, whereas the tail and arrowhead on an edge means that they occurred in all MAGs. Following is interpretation of edges [113], where the causal rank is from 0 to 4, and 0 being lowest and 4 being highest causality:

1. $X \rightarrow Z$: both X and Z are conditionally dependent, and X is a cause of Z. This is the strongest relationship of all, and we have assigned this the highest rank 4;

2. $X\ o\!\rightarrow Z$: the algorithm is certain that Z is not the cause of X (but not other way around) as this relationship was found in all MAGs. We have assigned this a rank 3;

3. $X\ o\!-\!o\ Z$: the algorithm is uncertain about whether X causes Z or Z causes X. This uncertainty occurred because both of these relationships were found in different MAGs. We have assigned this a rank 2; and

4. $X \leftrightarrow Z$: the bi-direction indicates that this edge was influenced by hidden variable, and X and Z have a common

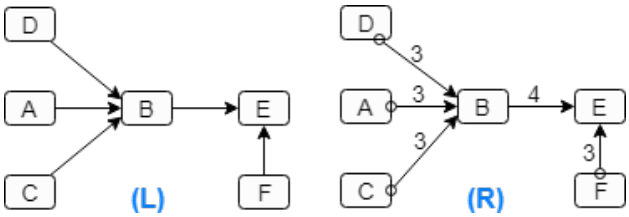

Figure 3: (L) DAG created from the temporal-association rules given in Table 1 and (R) a PAG created from DAG where each edge is assigned a causal rank, using the FCI algorithm.

cause. However neither X causes Z nor Z causes X, they are spouses. We have assigned this a rank 1 due to the lowest form of causality;

5. All other associations, where the FCI algorithm assigned undirected edges ($-$) or no edge at all, are given rank 0. The reason being FCI did not find any evidence of causal connection.

Our solutions uses the R package (pcalg) to obtain the PAG in the form of an adjacency matrix, in which circle is denoted by 1, arrowhead by 2 and tail by 3 [114]. For example, to represent $X \rightarrow Z$ in terms of adjacency matrix, the algorithm will insert 3 at X^{th} row and Y^{th} column, 2 at Y^{th} row and X^{th} column. Using the adjacency matrix of PAG generated by FCI algorithm, all (directed) event correlations in DAG are assigned corresponding causality ranks based on the aforementioned criteria. Our technique is somewhat similar to Linear Non-Gaussian Acyclic Model [115], which performs independent component analysis to develop linear causal models by pruning edges in the DAG. However, instead of removing the relationships, our solutions assigns a causal rank to signify the strength of causal relationship. This conversion of correlation into causal rules provides an additional layer information in terms of reliability and confidence. In the remainder of this paper, rules are termed as temporal-association-causal rules.

Continuing the example from Table 1, the extracted DAG is shown in Figure 3 (L), whereas a resultant PAG along with an assignment of causal rank on each edge is shown in Figure 3 (R). The PAG represents the temporal-association-causal rules and has 6 vertices (A, B, C, D, E and F) and 5 edges. Notice that the vertices represent the events and the edges are the temporal-associations. The ($o\rightarrow$) edge between D, A and C and B shows that B is not the cause of D, A and C, however the algorithm is not certain if D, A and C cause B. Due to this uncertainty, the assigned causal rank is 3. Similarly ($Fo\rightarrow E$) shows that E is not the cause of F, but there is no evidence that F causes E. At the point, the vertices D, A, C and F have been observed, where B is conditionally dependent on D, A and C and E is conditionally dependent on F. Given these dependence conditions, the algorithm declares that B is the cause of E, and is assigned a causal rank of 4. In other words, if the conditionally independent events D, A, C and F have occurred regardless of the order, then B is the cause event, whereas E is the effect event. This categorical separation of events, which becomes known due to the different causal ranks of temporal-association rules, helps in distinguishing between the (user) actions and (administrative) responses in terms of security.

Now the temporal-association-causal (TAC) rules have been established and stored in a simple database. The database acts as a central repository of rules, and further allows the exploration of a larger knowledge-base from multiple sources for a given problem. Each rule consists of one cause and one effect event, and has three associated values: number of times a rule was found in different datasets (counter), temporal-association accuracy (TAA) and causal rank. When new set of rules are extracted from an unseen dataset, the new and existing TAC rules go through the process of filtration again. We realise this is a resource intensive task, but it is necessary to always keep updated rules and values, which are free from cycles, redundancies and conflicting rules. In case a mismatch is found between new and existing rule, the rule with higher set of values will be stored in the database and the rest will either be updated or removed. The values are calculated as follows:

- *Counter* – When a new rule is inserted in the database, the counter starts at one. After processing a new dataset, if the solution finds one or more rules that already exist in the database, the corresponding counters of rules are increased by one. The counter value depicts the persistence of a rule, i.e., if a certain rule occurs multiple times, there is more chance of it to be true;

- *Temporal-association-accuracy (TAA)* – This value is calculated by Algorithm 1; and

- *Causal Rank* – This value is calculated by FCI algorithm as described in the current section;

7. Knowledge Representation and Utilisation

Utilising the generated temporal-association-causal (TAC) rules on a previously unseen machine is not feasible as algorithmic support is required. As the problem of selecting rules to use is a deliberation problem, we consider the use of an automated deliberation techniques in the form of automated planning (AP) [28]. The TAC rules present a collective set of expert actions that can be applied to any machine. However, it is unknown which actions will serve the purpose, i.e., only a subset of actions might be applicable and useful. Furthermore, there is no certainty that the order of acquired actions is same as the order that would be applied on an unseen system. Therefore, a clear decision-making process is needed regarding which actions to select for any given machine. This deliberation is traditionally performed by a human expert, which requires a significant time and effort. In this work, we consider the use of AP to replace manual effort. Hence the final stage of the research is to model the TAC rules into discrete actions with a precondition and effect, and utilise AP techniques to increase the usability of the proposed solution.

AP algorithms implement a state-transition system and is a 3-tuple $\sum = (S, A, \rightarrow)$, where $S = (s_1, s_2, \dots)$ is a finite set of states, $A = (a_1, a_2, \dots)$ is a finite set of actions, and $\rightarrow$: $S \times A \rightarrow 2^s$ is a state-transition function. A solution P is a sequence of actions $(a_1, a_2, \dots, a_k)$ corresponding to a sequence of state transitions $(s_1, s_2, \dots, s_k)$ such that $s_1 =\!\!\rightarrow (s_0, a_1)$, $\dots$, $s_k =\!\!\rightarrow (s_{k-1}, a_k)$, and s_k is the goal state. A system's configuration is represented by a set of first-order predicates, which are subsequently modified through the execution of each action, $a = \{pre+, pre-, eff+, eff-\}$, where $pre+$ and $pre-$ are the first-order preconditions that are positive and negative in the action's precondition list, respectively. Similarly, $eff+$ and $eff-$ are the action's positive and negative effects.

The proposed solution uses Planning Domain Definition Language (PDDL), which is an official language of International Planning Competition (IPC), to represent TAC rules based on state variables [116]. Having PDDL as a common format for representation allows better knowledge distribution and more direct comparison of systems and approaches. An important feature of PDDL is the division of planning in two parts; (1) domain model that only describes the knowledge and (2) a stand-alone problem instance that needs to be solved. This ability allows a categorical separation of elements belonging to domain and problem representations. The PDDL representation of extracted rules enables the proposed solution to utilise AP algorithms to produce a plan of actions.

7.1. Domain Modelling

A PDDL domain model consists of types of objects, predicates modelling facts, functions to handle numeric functions and actions. Objects and predicates are used to model the underlying system and an action models an evaluation or configuration step to increase the security. A precondition represents at what point the action is selectable, and an effect models a change to the objects, predicates and functions. The cause becomes the precondition, whilst the effect event of a rule becomes the effect part of an action. This is due to the reason that cause lead to the effect event and they are also ordered in term of time. The name of an action is selected as 'LHS event-to-RHS event'. The combined list of object names from both cause and effect of a rule constitutes as parameters. Each domain action having one or more parameters will provide all objects that are required to make any configuration change in the underlying system. It should be noticed here that the domain actions are same as the TAC rules.

A predicate is placed in each action's effect to avoid the successive selection of the same actions, which involve the same objects. It is false before the action execution and means that the given objects have not been processed yet, and consequently updated true to depict that the same objects cannot be executed again. For example, `(not (e_1234 ?parameter1 ?parameter2)` is used to prevent the same objects of `?parameter1` and `?parameter2` appearing again with `e_1234`. Note that this predicate would not restrict the repetition of actions, it would just prevent redundancy that can occur due to absence of goal state and use of maximisation metric in a problem instance (discussed later in Section 7.2). This way the planner would not duplicate the same plan of actions repeatedly to attain maximum TAA value and provide a precise set of actions.

A function, named `accumulative-weight`, is used in the domain model to hold the amount of certainty or importance of any extracted rule. The accumulative-weight value is the product of the counter, TAA and causal rank values. Therefore, higher accumulative-weight values indicate stronger relationships. A TAC rule (4720 $\rightarrow$ 4732) converted into a PDDL action with an accumulative-weight of 104 is shown in Figure 4. The event type 4720 shows

```
(:action e_4720-to-e_4732

   :parameters (?DisplayName ?HomeDirectory ?HomePath ?PasswordLastSet ?SamAccountName
      ··· ?TargetDomainName ?TargetSid ?TargetUserName ?UserWorkstations ?MemberSid)

   :precondition (and (e_4720 ?DisplayName ?HomeDirectory ?PasswordLastSet ?ScriptPath
      ··· ?TargetDomainName ?TargetSid ?TargetUserName ?UserWorkstations ?MemberSid) )

   :effect (and
      (increase (accumulative-weight) 104.003)
      (not (e_4720 ?DisplayName ?HomeDirectory ?PasswordLastSet ?ScriptPath ?TargetDomainName
         ··· ?TargetSid ?TargetUserName ?UserWorkstations ?MemberSid))
      (e_4732 ?SubjectDomainName ?SubjectLogonId ?SubjectUserName ?SubjectUserSid ?MemberSid
         ?TargetDomainName ?TargetSid ?TargetUserName))
)
```

Figure 4: An example PDDL action, which is created using temporal-association-causal relationship of two events. It also shows the relevant objects/parameters and an accumulative-weight value.

that a user account was created, whereas 4732 informs that the account was added to a security enabled local group. The objects and parameters involved in event were employed to carry out the configuration actions. The objects distinctively represent the occurrence of certain event, and are necessary for non-experts to interpret the plan actions into an actionable knowledge.

7.2. Problem instances

Problem instances are automatically constructed using a given domain action model and event log entries of an underlying machine. The problem instance contains objects, initial state, goal state and a metric. The use of domain model ensures that the object types and initial state of problem instance are limited to the types and predicates of domain model to prevent any execution errors during the planning. The objects list and their corresponding types are extracted from event log entries using object or property name-value pairs. The goal state is kept empty as it is not known what actions need to be scheduled and there is no way of finding the missing configuration in target machine. To obtain the complete set of actions or those with the highest accumulative-weight value in the available planning time, the maximisation metric is applied in the problem file. The PDDL does not allow multiple metrics in a single instance, hence the reason for using the product of counter, TAA and causal rank values as accumulative-weight. The complete set of actions provided against the identified issues based on the knowledge encoded in domain model ensure to aid the non-experts to identify the security risks. The total `accumulative-weight` value of a plan is incremented by the `accumulative-weight` of each chosen action. A live system problem file is shown in Figure 5, which was extracted from a machine having poor security configurations. The objects and current security state in the form of event types are extracted from the log entries of the machine. For example, `Test1 - DisplayName` represents an object `Test1`, which is of `DisplayName` type. Same goes for the `WORKGROUP - SubjectDomainName`, where domain name of underlying subject is `WORKGOUP`. The `accumulative-weight` value is initialised with zero and will be incremented (to maximum extent) according to the execution of domain actions.

7.3. Plan generation

The Local search for Planning Graphs (LPG) [117] planner is used due to its good performance and support for PDDL [118]. The planner was executed in incremental mode to identify plans of increasing quality within a specified time. Using domain and problem files as input to LPG planner, the plan is extracted that explains the set of actions to improve the security of underlying machine. The order of plan actions is important to perform the configuration tasks. For sake of an example, a small segment of plan extracted from a live system with limited set of objects is shown in Figure 6. The three actions describe the process of creating a secure account in a networked environment. The objects

```
(define (problem problem_Events)
   (:domain EventRelations)
   (:objects
      Test1 - DisplayName                          0_1794 - PasswordLastSet
      WORKGROUP - SubjectDomainName                IE8Win7 - TargetDomainName
      0_0x3e7 0_0x105f5 - SubjectLogonId            IE8SAAD$ IEUser1 - SubjectUserName
      S_1_5_18 S_1_5_19 - SubjectUserSid            Administrators - TargetUserName
      S_1_5_32 - TargetSid                         S_1_5_20 - MemberSid
      SeTcbPrivilege SeCreateGlobalPrivilege - PrivilegeList
      ⋮
   )
   (:init
      (= (accumulative-weight) 0)
      (e_4720 Test1 0_1794 WORKGROUP 0_0x105f5 IE8SAAD$ S_1_5_18 IE8Win7 S_1_5_32 ···) )
      ⋮
   (:goal (and ) )
   (:metric maximize (accumulative-weight))
)
```

Figure 5: A simplified version of problem file generated automatically from a machine that has poor security configurations.

```
(E_4720-TO-E_4732 Test1 WORKGROUP 0_0x105f5 IE8SAAD$ S_1_5_18 IE8Win7 S_1_5_32 Administrators ···)
(E_4732-TO-E_4728 S_1_5_20 WORKGROUP 0_0x105f5 IE8SAAD$ S_1_5_18 IE8Win7 S_1_5_32 Administrators)
(E_4728-TO-E_4722 S_1_5_20 WORKGROUP 0_0x105f5 IE8SAAD$ S_1_5_18 IE8Win7 S_1_5_32 Administrators)
⋮
```

Figure 6: A small segment of plan for the target machine using the domain knowledge acquired in an automated manner.

of each action describe the information needed to perform the action. The object names are given in Figure 5. The event type 4720 shows that a user account `IE8SAAD$` was created in a network by another account `Administrators`. After that, `Administrators` added the `IE8SAAD$` account to local and global security groups as evident by event 4732 and 4728, respectively. At the end, event type 4722 shows that the account `IE8SAAD$` was enabled. Although it is a small segment of a plan, these actions simply inform the non-experts to assign appropriate user-group when creating a new user account for better security.

The plan only includes those actions from the domain model, which were selected by the planner based on matching the initial state of the problem instance. The matched initial state of a plan describes the security issue or a missing configuration, while the remaining part manifests the mitigation actions. As the routine events are filtered in early stages and assuming the domain model comprises of relevant and sufficient knowledge, the actions in a plan can perform one of the following tasks: (1) recognise a pattern of a security issue and (2) identify a security issue and propose a method to resolve it as well. It is also possible that the plan delivers partial information; however, it would still be useful to some extent for the non-experts as discussed in later sections. Furthermore, the objects of each action in a plan play an important role in conveying the desired level of information as they belong to the vulnerable machine.

7.4. Benefits of AP application

The application of AP is integral to the proposed solution due to following advantages:

1. AP algorithms provide systematic deliberation mechanisms to replace that of the human expert;
2. The parameters of domain actions provide additional information on the specified TAC rule;
3. Objects in the actions of a plan belong to the underlying machine, hence providing usable results;
4. Ability to optimise based on a metric, ensuring that actions with the highest accumulative-weight values are selected to output accurate results; and
5. The knowledge is saved in a standardised format and can be utilised by any PDDL supporting planner, and thus provides a benchmark domain for the AP community.

7.5. Summary

To get a better understanding of the overall solution, Figure 7 summarises the steps taken from processing the event logs to producing a security action plan. The proposed solution consists of two parts; gathering the expert security knowledge and applying it to previously unseen, vulnerable machines. The first part begins by reading the given event logs, filtering the routine events and extracting the security knowledge in the form of Temporal-Association-Causal (TAC) rules. Each TAC rule is then stored in the database and contains the following information: properties of every event in the form of object-value pairs, number of times this specific rule was found in different datasets, a temporal accuracy value and a causal rank. In the second part, every rule in the database is used to create a domain action model using PDDL as well as generating a problem instance from the event log entries of a vulnerable machine. After that, LPG planner is used to generate a plan solution, which is then parsed to represent the identified security risks along with the mitigation actions.

8. Empirical Analysis

This section presents an empirical analysis of the proposed solution using the security event logs acquired from 21 live machines. Here 'live' is used to denote that they are computing devices used in the real-world and not solely for research purposes. The main purpose is to test if the automated solution was successfully able to extract knowledge from the 21 test machines, and determine the effectiveness and productivity against a manual log analysis approach, which requires expert knowledge, time and effort. The complete source code, event log datasets and results are available upon request. The reason behind including multiple machines to perform the evaluation is to verify that the proposed solution is sufficiently adaptable and usable under various circumstances. The accuracy of the solution is based on its ability to successfully propose human-like actions. The adopted methodology has already been used to test several automated domain learning approaches [119, 120].

8.1. Methodology

Following explains the procedure of conducting empirical analysis of the proposed solution on real-time machines.

8.1.1. Data collection

To acquire test data for the solution, 21 machines are selected that are part of a university network and used on a daily basis. All machines are configured by administrators according to the pre-defined security policies of the university. These machines have different operational security roles/settings (e.g. student, staff, administrator etc.) and have distinct characteristics, such as number of routine events, unique events etc. As the security policies are known, their interpretation by the experts provided the ground truth and made it possible to determine the accuracy of proposed solution.

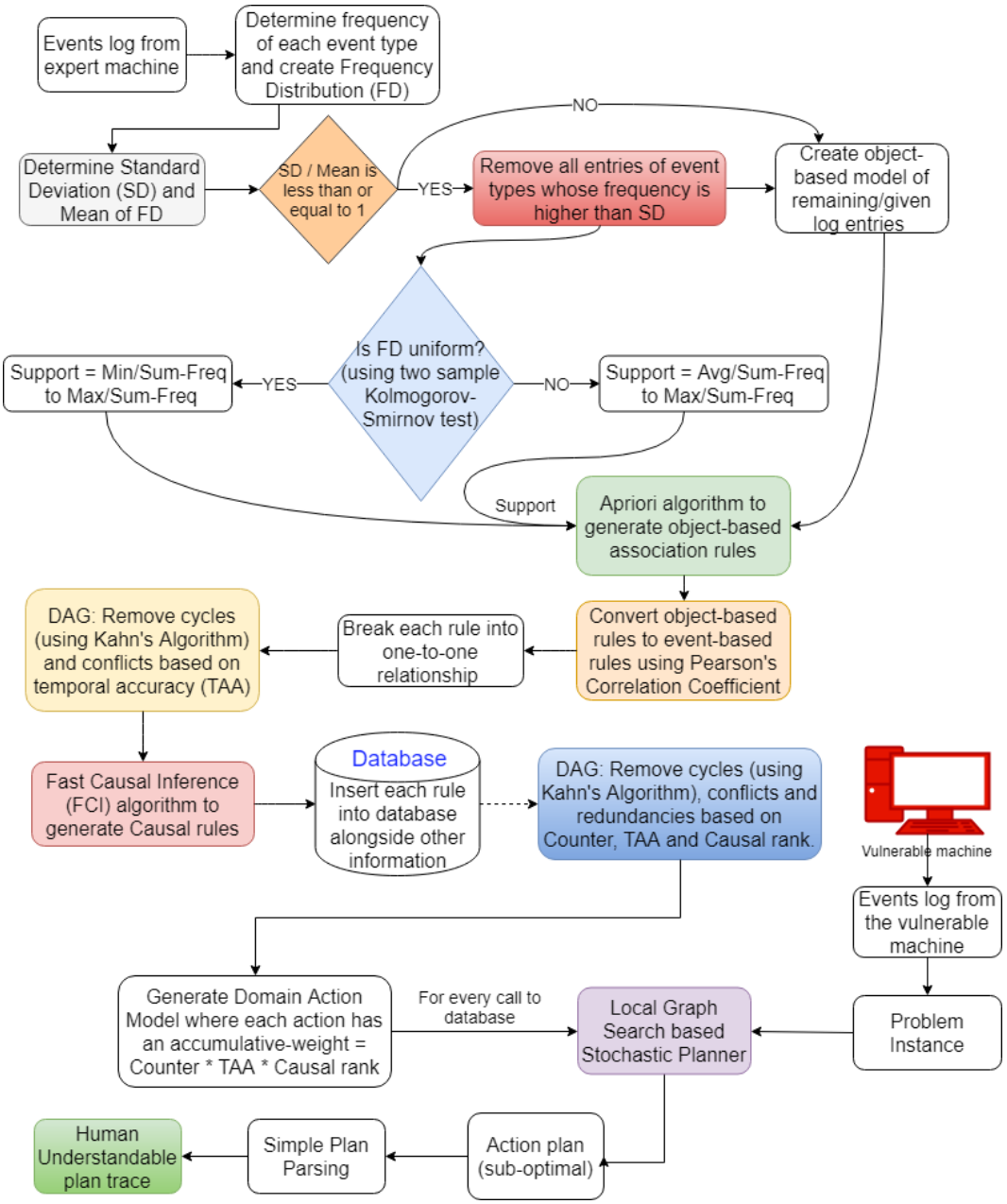

Figure 7: Overall process of the proposed solution.

8.1.2. Automated plan generation

The first step is to extract the security event logs of 21 machines, which as we are using Microsoft machines are in the form of '.evtx' files. The second step is to process each file using the automated tool to generate TAC rules of event relationships, which are further encoded into respective domain action models. The third step is to create a problem instance from a vulnerable machine, which is essentially an entirely unconfigured machine containing a new copy of Microsoft Windows OS and requires various security improvements. The reason behind selecting such a machine is to test the developed solution's capability under adverse circumstances, and determine if it can recognise all security issues appropriately. Finally, using the acquired domain models and the problem instance, 21 plans are produced using the Local search for Planning Graphs (LPG) planner to test the accuracy of the solution that is based

on the ability to resolve security issues of the machine.

8.1.3. Manual plan generation

Using the same set of security event logs, 21 domain action models are manually produced, where each model is hand-crafted by three security auditors of the university. All auditors are experts in their fields, and have day-to-day jobs to monitor event logs and identify interesting patterns in terms of user activities. The reason behind using multiple experts is to perform fair and unbaised analysis. A manual domain model represents the set of actions that were taken by a human for either configuration or to resolve a particular issue in an underlying machine. Processing a number of datasets, each with large number of entries, can be a time-consuming and error prone task. To overcome this issue, we developed an assistive software tool where the experts can view and link all events (types, descriptions and objects) of a given dataset, and generate a domain action model without manually encoding it.

Using the manually constructed domain models and the same problem instance, 21 plans were produced. Each plan depicts the actions that human experts would take to resolve security issues of the same vulnerable machine. The manual plan is created using a combined knowledge of ground truth (i.e. security policies of the 21 machines) and expert opinions with respect to the security issues. Note that the large amount of time required to generate a manual set of security actions further motivates the need for our proposed automated solution, where the complete process is performed without human intervention.

8.1.4. Accuracy

The solution's accuracy is determined by comparing plans generated manually (created by human experts) and automatically (created by the proposed solution). The comparison identifies the number of correct, incorrect, additional and missing actions (event relationships). An action of automated plan would be considered correct if the same relationship, having identical events and direction, is found in the manual plan. Similarly, an action of an automated plan would be considered incorrect if the same events are found in the manual plan, but with an opposite relationship direction. An action would be considered as missing, if it exists in the manual plan, but is absent from the automated plan. Similarly, an action would be considered as additional, if it exists in the automated plan, but is not included in the manual plan. After finding these values, following formula is applied to calculate accuracy: $(correct/(correct + incorrect + missing + additional)) \times 100$. To simplify the process, we developed another tool that takes both manual and automated plans as input and outputs the accuracy value. Note that if the automated domain model is complete and accurate, the generated plans will represent similar steps that were taken by the expert(s) to perform a certain security-related task.

It is possible that the learned event patterns may be all obvious ones for security experts, but the novice users are unfamiliar with them. The complexity and novelty of the knowledge depicted by an automated plan depends on the given problem instance, i.e., security status of an underlying test machine. If the machine has poor security, the automated plan would be large and more complex, otherwise, it will contain relatively simple and easy-to-configure actions. Furthermore, the perception of complexity of the knowledge is subjective to the background and experience of the users. Hence, this empirical analysis only considers quantitative accuracy of the event patterns, rather than the qualitative accuracy. The accuracy measure does not consider how easy or difficult would it be for the user to implement the proposed security solution. It only examines whether the set of actions are capable of identifying the security issues and proposing a solution for the target machine.

8.2. Results and discussion

The empirical analysis presented in this paper was performed on a 32-bit Microsoft Windows 7 virtual machine with 2 cores of Intel i7 3.50GHz CPU and 4GB RAM. Table 3 presents the results from each layer of processing, such as number of event log entries, rules, plan, execution time, accuracy etc. It should be noticed that the table only mentions TAC rules but not the domain actions as both are the same. The acquired domain models from 21 datasets have presented suitable accuracy, ranging from $73\% - 92\%$. The solution has also shown reasonable performance, taking 39 minutes for processing 69,854 entries with 67 unique events and 31 minutes for processing 313,470 entries with 44 unique events. There is potential for a degree of uncertainty in finding the accuracy as the experts do not know the complete set of events generated when a particular security configuration activity was performed on a machine. Moreover there can be one or more ways of executing the same task, which will produce a different set of event

entries. For the same reasons, true and false negatives were not included in determining TAA value due to ambiguity in what constitutes a negative. There is also uncertainty around the plan as LPG is a sub-optimal planner, but its ability to apply stochastic search to draw plan without any goal state motivated the use. Furthermore, any successfully found plan will provide one of the many ways to conduct a certain configuration.

Event logs		*Rule Mining and Domain Modelling*								*Automated Planning*		
Event log dataset	***No. of events***	***Minimum Support***	***Maximum Support***	***No. of unique events***	***No. of object-based rules***	***No. of event-based rules***	***No. of temporal-association rules***	***No. of temporal-association-causal rules***	***Domain extraction time***	***No. of actions in plan***	***Planner execution time (s)***	***Plan Accuracy %***
1	5,027	0.07	0.59	29	516608	14	84	32	08m:03s	37	1.17	87
2	8,253	0.08	0.49	31	107503	5	19	11	04m:46s	30	0.89	85
3	521	0.25	0.49	16	147	6	11	8	15m:42s	4	0.04	83
4	1,122	0.05	0.53	20	266671	5	3	3	03m:21s	2	0.06	74
5	1,079	0.05	0.33	23	64931	9	14	10	01m:00s	3	0.02	88
6	1,691	0.05	0.20	27	263800	4	6	6	02m:58s	6	0.18	80
7	1,714	0.13	0.46	25	978	9	58	21	02m:53s	32	0.83	87
8	9,721	0.08	0.33	31	4252	8	11	9	04m:34s	6	0.17	83
9	9,770	0.05	0.32	29	4027	9	21	12	04m:37s	5	0.14	73
10	13,026	0.06	0.72	29	1176	8	8	7	07m:02s	4	0.9	85
11	10,948	0.05	0.62	13	1349	5	33	11	05m:29s	3	0.07	85
12	8,832	0.07	0.41	19	15729	4	12	10	05m:05s	4	0.16	90
13	3,403	0.05	0.37	32	97574	10	7	6	07m:00s	8	0.27	78
14	31,719	0.11	0.69	14	150593	3	13	9	26m:04s	49	1.24	82
15	1,072	0.07	0.26	18	14681	7	7	6	00m:47s	3	0.02	86
16	1,991	0.05	0.39	33	4959	17	160	75	01m:42s	348	5.38	91
17	826	0.05	0.40	23	280482	4	8	5	05m:01s	6	0.08	81
18	5,309	0.06	0.23	29	46151	5	4	4	03m:10s	3	0.07	75
19	69,854	0.05	0.37	67	66600	10	16	15	38m:33s	72	1.63	92
20	313,470	0.05	0.34	44	215	6	115	43	30m:45s	80	1.91	87
21	32,688	0.15	0.43	21	237155	4	6	6	27m:51s	7	0.09	80

Table 3: Results of empirical analysis from domain learning and automated planning. The analysis is performed on 21 event log datasets from live machines and shows the output numbers from each stage of processing.

The higher amount of unique entries relative to total entries implies substantial user activity and fewer routine tasks. As the proposed solution is data-driven, this results in more number of domain actions. According to Table 3, dataset-16 has 1,991 total and 33 unique entries, dataset-19 has 69,854 total and 67 unique entries and dataset-20 has 313,470 total and 44 unique entries. Among these 3 datasets, dataset-16 produced the highest number of temporal-association-causal rules (75) and plan actions (348) in under 2 minutes. On the contrary, the minimum number of unique entries (13) relative to total entries (10,948) were found in dataset-11, which only produced 11 TAC rules and 3 actions in plan. This indicates that the dataset has relatively large number of routine events, which have occurred repeatedly over time. Due to this excessive filtering, dataset-11 required 5.5 minutes to process. The aforementioned discussion shows that the higher number of unique entries against total entries produced a larger object-based model, and therefore resulted in more number of domain actions. Hence, this proportional relationship between the number of unique events and domain actions demonstrates the completeness of the proposed solution.

The number of object-based association rules, final Temporal-Association-Causal (TAC) rules and domain extraction time are all independent of the number of input log entries. The maximum number of association rules (516,608) is generated from dataset-1, which only has 5,027 total entries, and resulted in 32 TAC rules in 8 minutes. Similarly dataset-6 produced 263,800 association rules, but only resulted in 6 TAC rules. On the other hand, dataset-16 only used 4,959 association rules to output 75 TAC rules in less than 2 minutes. Due to the temporal validation of association rules and formation of Partial Ancestral Graph (PAG), the majority of false positive rules are either corrected or removed, and the solution generates useful results in feasible time. Furthermore the elimination of unwanted rules on each phase reduces the overall processing time. This signifies the efficiency of the proposed solution.

Another important thing to consider is the difference between the number of temporal-association and TAC rules. The difference is due to the creation of DAG, which eliminates all redundancies, conflicts and cycles. It also depends on the characteristics of dataset and the extracted temporal-association rules. For dataset-1, 52 rules were filtered. For dataset-2, 8 rules were filtered and so on. Each TAC rule is given a causal rank, which either increases or decreases its overall accumulative-weight value. Hence, besides TAA and counter values, the causal rank also enforces the planner to select the most feasible actions (from entire domain knowledge) in a plan. This signifies the importance of application of causality that lead to better accuracy. However in case of dataset-4, 6, 18 and 21, the number of temporal-association and TAC rules are equal, which shows that correlation can imply causality but that is not always the case.

It has also been observed that datasets with comparatively lower accuracy contain such rules, where one event is linked to many others. Here Automated Planning plays an important role to determine the suitable set of actions with respect to accumulative-weight values. Another observation is that the solution does not force event relationships, where they are not present, regardless of number of entries in a dataset. These datasets usually contain a large number of entries, but a few unique event types. Some of the datasets were excluded from the testing as they did not produce any domain actions. The object-based model ensures that a rule, and consequently a domain action, is created only where there is a relation among the properties of event log entries. Apart from the results, it is important to consider that the solution will provide improved accuracy if the configuration, environment and intended use of target computer are same as of the machine from which the knowledge was extracted. The similar settings and context will make it easier to match any learnt knowledge and help the user perform any necessary changes.

8.3. Comparison of manual and automated plans

This section presents a comparison between an automated and manual plan, and demonstrates the ability of the proposed solution to suggest human-like actions. Both complete plans, including security identification and mitigation actions, are shown in Table 4, whereas a detailed elaboration of the automated plan actions is given in Section 8.4. The actions of a plan can be difficult for the non-expert to understand. Hence, we created a parsing tool that uses event log entries of the vulnerable machine and the domain action model to describe the security issue and respective solution, along with elaborating each action of the plan. The tool explains all related events and locates corresponding object name to each of the parameters. The accuracy calculation mechanism covers all aspects by finding the number of correct, incorrect, additional and missing actions from any automated plan. For the current automated plan, all 8 actions are found in the manual plan and there are no incorrect and additional actions. However, the automated plan is missing three actions (5, 6 and 7) that are present in the manual plane. The missing actions are: 4798 $\rightarrow$ 4728, 4728 $\rightarrow$ 4738 and 4738 $\rightarrow$ 4647. Hence the accuracy would be: $(8/(8 + 0 + 3 + 0)) \times 100 = 73\%$.

Automated plan		Manual plan
Actions	*Relevant Objects*	*Actions*
1. 5145 → 5140	Share name: `Users`, Subject: `IEUser1`, SID: `S-1-5-18`	1. 5145 → 5140
2. 5140 → 4648	Target-domain: `IE8Win7`, server: `localhost`	2. 5140 → 4648
3. 4648 → 4657	Operation: `registry value added`, Type:`Mult-String`	3. 4648 → 4657
4. 4657 → 4798	Subject: `IEUser1`, Domain: `WORKGROUP`, Process: `mmc`	4. 4657 → 4798
5. 4798 → 4647	Subject: `IEUser1`	5. 4798 → 4728
6. 4647 → 4624	Subject: `IEUser1`, SID: `S-1-5-19`, Domain: `WORKGROUP`	6. 4728 → 4738
7. 4624 → 4726	Subject: `IEUser1`, Domain: `WORKGROUP`	7. 4738 → 4647
8. 4726 → 4729	Target: `IEUser1`, SID: `S-1-5-19`, Subject: `Administrator`	8. 4647 → 4624
-	-	9. 4624 → 4726
-	-	10. 4726 → 4729

Table 4: Depiction of automated and manual plans alongside relevant objects. The conversion of actions of plan in a user-friendly format is performed by the plan parser.

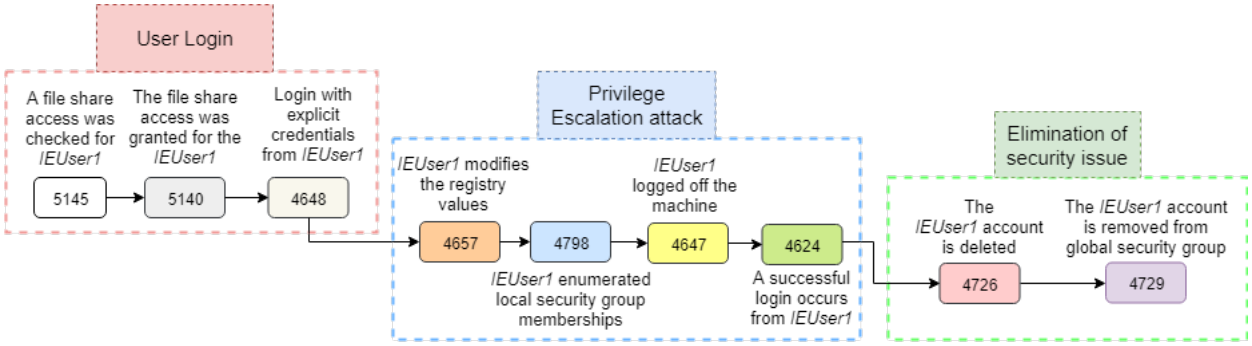

Figure 8: A complete timeline of events extracted in the form of plan. The related events and corresponding objects reflect the actions of security expert(s) on a particular machine. The red and blue coloured boxes show the events found in vulnerable machine, while the green coloured box shows the mitigation plan.

8.4. Interpretation of a plan

A timeline demonstrating the complete security concern and mitigation (from Table 4) is presented in Figure 8. The timeline consists of the full list of actions included in the plan; however, it is important to communicate that a large portion of the actions have already been discovered in the event log of a vulnerable machine. This section also demonstrates that the proposed solution is easy to use and an automated plan only requires a basic knowledge for interpretation. It should be noticed here that the sequences of events generated from a single dataset are not universally applicable. The event entries are logged as soon as they are generated by multiple system activities, and there is no way of knowing which of them are connected with respect to a single expert action. The solution presented in this paper can be used to find those sequences and connections in the presence of large number of routine events, and facilitate the autonomous replication of human expert actions into a domain model. Following is the brief interpretation of planner output:

- ***Matched*** – This part of the plan shows the security issue of the vulnerable machine and was identified by matching one or more initial state(s) of the problem instance with the domain action model:
  - *User Logon* – Event 5145 shows that a certain user, with a `S-1-5-18` group security identifier (SID) and `IEUser1` username, tried to access file-share service on a remote server. The group SID is a unique value of variable length used to identify a trustee and issued by an authority. The event 5140 shows that server access was allowed for `IEUser1`. After that, the remote server requested the login credentials from `IEUser1`. The required username and password were provided to the server as evident by event 4648; and

```
(E_5058-TO-E_5061 RSA d5_0e7b98a02239 SMS O_2499 AD O_0x3e7 CES000963029$ S_1_5_18 ···)
(E_5061-TO-E_5441 RSA d5_0e7b98a02239 O_16390 GUID_MFE_TCP_STREAM_CALLOUT_V4 DataFinder ···)
(E_5441-TO-E_4625 Kerberos AD O_0x3e7 CES000963029$ S_1_5_18 S_1_0_0 ···)
```

Figure 9: Incorrect set of actions extracted for the test machine using the acquired domain knowledge.

- *Privilege Escalation attack* – Following the login, event 4657 is triggered which shows that the `IEUser1` added a new registry value. After that, event 4798 informs that the `IEUser1` gathered local users of the server. The next two events (4647 and 4624) reveal that the `IEUser1` logged out of the server, and then successfully logged back in, but, with a different group SID `S_1_5_19`. It should be noticed here that both events 4648 and 4624 indicate the user account login, however the event 4648 shows that the login occurred through 'RunAs' command or some local program.

- ***Partially Matched*** – This part of the plan provides the security solution, which was discovered due to the propagation of linked domain actions:

  - *Elimination of security issue* – The next event 4726 shows that the `IEUser1` account was deleted from the remote server. The last event 4729 further informs that the account was removed from global security group by a server administrator.

Based on the sequence of events, it shows that an administrator found a suspicious activity in terms of system registry modification by a local user account (malicious insider), which lead to the discovery of possible privilege escalation attack on a remote server. At this point, it is unknown whether the account became an actual member of another user-group, but it became clear later when the security identifier (SID) was changed from `S_1_5_18` to `S_1_5_19` upon a second login. The user might have exploited a server vulnerability (possibly by introducing a virus) to login the server and added the registry value, which made it possible to identify other users. This process enabled the user to find and change to a privileged user-group. Furthermore, the plan shows that the new value added in the registry was a string, which could be a 'net' command to add the current user in administrators group, and placed in *HKLM\Software\Microsoft\Windows\currentversion\run*. As the user initially accessed file-share service of the server, the new user-group might have higher access permission where the service could be compromised. The administrator resolved this issue by completely removing the user-account from the server.

Continuing the discussion from Section 8.3, the automated plan is missing information as it is not fully accurate (only 73%). The missing information can be found in the manual plans. As evident by this sequence (4798 $\rightarrow$ 4728 $\rightarrow$ 4738 $\rightarrow$ 4647), after gathering the user-groups (event 4798), the account of `IEUser1` was actually added to a security-enabled global group (event 4728). This change in the user account `IEUser1` triggered the event 4738. Later on, `IEUser1` logged off from the server (event 4647).

There is also a potential of generating an incorrect plan, where the extracted sequence of actions present erroneous method of conducting security-related activities. This happens due to wrong event relationships. An example of incorrect plan is presented in Figure 9. At first, the events 5058 and 5061 show that a cryptographic operation was performed on a Key Storage Provider file. The operation can be one of: create, delete, export, import, open and store cryptographic keys. The next event is 5441, which is recorded for every filter of Windows Filtering Platform at every system start-up. This event just documents the time of startup and does not indicate any configuration change. The last event indicates a user-account failed to log on by providing a bad username or password. This plan is wrong because it has linked two separate system activities, i.e., key operation of Cryptography and Windows logon service, with each other. The order of events is also incorrect as all cryptographic operations are performed after the user login. Furthermore, these actions do not provide any useful or actionable knowledge to the non-experts that might improve their systems' security.

8.5. Application in other information systems

To apply the proposed technique on other operating systems and applications (information systems), there would be a need to create parsing tools or software modules that can process the respective event log datasets to generate object-based models as described in Section 3. The event log entries are found in different text-formats, and requires different parsing and analysis mechanisms to obtain objects. Doing this will increase the applicability and adoption of the proposed solution.

Furthermore, there are some event logging mechanisms, e.g. application-specific security event logs, which do not assign a numeric identifier to an event. In general, it is considered good practice to add a unique identifier for every event type as it increases the parsing accuracy [121]. A small amount of research has been done that automatically introduces event identifiers in the raw event entries [122]. Extending and incorporating such software modules in the developed solution will also lead towards wider adoption.

9. Conclusion

This paper presents a novel automated technique to extract a domain action model from a security event log without any human intervention. The main purpose of this solution is to enable non-experts to conduct expert analysis of the new or unseen machine without spending significant amount of time and effort in acquiring security knowledge. The technique is based on a scenario where one or more experts perform the security evaluation or configuration on a system. Every change or action will be recorded in the form of event log entries. Identifying temporal-association-causal relationships among such event log entries in an automated manner can provide a sequence of actions that the experts took to reform the system security. In addition, storing and representing the extracted knowledge in a standardised PDDL format will increase the applicability of the solution due to the wide presence and understanding of automated planners.

The presented technique was developed and tested using 21 event log datasets. First, it creates an object-based model to represent event log entries and extracts strong correlation rules using ARM techniques. The correlation rules are then formulated into sequences of temporally-associated rules using a temporal metric. The temporal-association rules are further processed and assigned a causal rank using the FCI algorithm. Knowledge extracted in the form of temporal-association-causal rules is encoded into a domain action model using PDDL. The developed solution automatically generates problem instances as well by encoding live events and objects from underlying machine. Given the domain and problem files, LPG planner is used to extract relevant actions in the form of a plan. Despite the event log entries are produced in high frequency and might contain large amount noise (routine events), the proposed solution demonstrated that it can successfully extract the domain knowledge with reasonable accuracy and be applicable in practical environments. Future work includes the augmentation of post-plan-processing to translate the output sequence of events into a step-by-step guide for the users. This will involve the development of tools and techniques that can reduce the manual effort in identifying and implementing security measures.

References

[1] V. Viduto, C. Maple, W. Huang, D. LóPez-PeréZ, A novel risk assessment and o model for a multi-objective network security countermeasure selection problem, Decision Support Systems 53 (3) (2012) 599–610.

[2] T.-D. B. Le, D. Lo, Deep specification mining, in: Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, ACM, 2018, pp. 106–117.

[3] T.-D. B. Le, X.-B. D. Le, D. Lo, I. Beschastnikh, Synergizing specification miners through model fissions and fusions (t), in: Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on, IEEE, 2015, pp. 115–125.

[4] L. Mariani, M. Pezzè, M. Santoro, Gk-tail+ an efficient approach to learn software models, IEEE Transactions on Software Engineering 43 (8) (2017) 715–738.

[5] A. Wasylkowski, A. Zeller, Mining temporal specifications from object usage, Automated Software Engineering 18 (3-4) (2011) 263–292.

[6] D. Schauland, D. Jacobs, Managing the windows event log, in: Troubleshooting Windows Server with PowerShell, Springer, 2016, pp. 17–33.

[7] C. Simache, M. Kaâniche, A. Saidane, Event log based dependability analysis of windows nt and 2k systems, in: Dependable Computing, 2002. Proceedings. 2002 Pacific Rim International Symposium on, IEEE, 2002, pp. 311–315.

[8] M. F. Mushtaq, U. Akram, I. Khan, S. N. Khan, A. Shahzad, A. Ullah, Cloud computing environment and security challenges: A review, INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS 8 (10) (2017) 183–195.

[9] S. Khan, S. Parkinson, Y. Qin, Fog computing security: a review of current applications and security solutions, Journal of Cloud Computing 6 (1) (2017) 19. doi:10.1186/s13677-017-0090-3.
URL https://doi.org/10.1186/s13677-017-0090-3
[10] D. Luckham, The power of events: An introduction to complex event processing in distributed enterprise systems, in: International Workshop on Rules and Rule Markup Languages for the Semantic Web, Springer, 2008, pp. 3–3.
[11] R. Vaarandi, Mining event logs with slct and loghound, in: Network Operations and Management Symposium, 2008. NOMS 2008. IEEE, IEEE, 2008, pp. 1071–1074.
[12] J. Myers, M. R. Grimaila, R. F. Mills, Log-based distributed security event detection using simple event correlator, in: System Sciences (HICSS), 2011 44th Hawaii International Conference on, IEEE, 2011, pp. 1–7.
[13] C. Liu, A. Singhal, D. Wijesekera, A model towards using evidence from security events for network attack analysis., in: WOSIS, 2014, pp. 83–95.
[14] P. P. Angelov, Autonomous learning systems: from data streams to knowledge in real-time, 1st Edition, John Wiley & Sons, 2013, pp. 133–144.
[15] J. Pearl, et al., Causal inference in statistics: An overview, Statistics surveys 3 (2009) 96–146.
[16] R. Vaarandi, Sec-a lightweight event correlation tool, in: IP Operations and Management, 2002 IEEE Workshop on, IEEE, 2002, pp. 111–115.
[17] A. Agarwal, D. Ahrens, R. Livingood, M. Mani, N. Singh, A. Zmolek, Multi-tier security event correlation and mitigation, uS Patent App. 12/234,248 (Oct. 8 2009).
[18] R. K. Sahoo, A. J. Oliner, I. Rish, M. Gupta, J. E. Moreira, S. Ma, R. Vilalta, A. Sivasubramaniam, Critical event prediction for proactive management in large-scale computer clusters, in: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2003, pp. 426–435.
[19] M. S. Boddy, J. Gohde, T. Haigh, S. A. Harp, Course of action generation for cyber security using classical planning., in: International Conference on Automated Planning and Scheduling (ICAPS), 2005, pp. 12–21.
[20] T. Vaquero, R. Tonaco, G. Costa, F. Tonidandel, J. R. Silva, J. C. Beck, itsimple4. 0: Enhancing the modeling experience of planning problems, in: System Demonstration–Proceedings of the 22nd International Conference on Automated Planning & Scheduling (ICAPS-12), 2012, pp. 11–14.
[21] R. M. Simpson, T. L. McCluskey, W. Zhao, R. S. Aylett, C. Doniat, Gipo: an integrated graphical tool to support knowledge engineering in ai planning, in: Sixth European Conference on Planning, 2014.
[22] T. L. McCluskey, S. Cresswell, N. E. Richardson, M. M. West, Action knowledge acquisition with opmaker2, in: International Conference on Agents and Artificial Intelligence, Springer, 2009, pp. 137–150.
[23] S. N. Cresswell, T. L. McCluskey, M. M. West, Acquiring planning domain models using locm, The Knowledge Engineering Review 28 (2) (2013) 195–213.
[24] A. Botea, M. Enzenberger, M. Müller, J. Schaeffer, Macro-ff: Improving ai planning with automatically learned macro-operators, Journal of Artificial Intelligence Research 24 (2005) 581–621.
[25] R. Jilani, A. Crampton, D. Kitchin, M. Vallati, Ascol: A tool for improving automatic planning domain model acquisition, in: Congress of the Italian Association for Artificial Intelligence, Springer, 2015, pp. 438–451.
[26] H. H. Zhuo, Crowdsourced action-model acquisition for planning., in: AAAI, 2015, pp. 3439–3446.
[27] K. Long, J. Radhakrishnan, R. Shah, A. Ram, Learning from human demonstrations for real-time case-based planning 31 (1) (2009) 1–6.
[28] M. Ghallab, D. Nau, P. Traverso, Automated Planning: theory and practice, Elsevier, 2004.
[29] R. A. Valenzano, N. Sturtevant, J. Schaeffer, K. Buro, A. Kishimoto, Simultaneously searching with multiple settings: An alternative to parameter tuning for suboptimal single-agent search algorithms, in: Third Annual Symposium on Combinatorial Search, 2010.
[30] S. Khan, S. Parkinson, Towards automated vulnerability assessment (2017) 33–44.
[31] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, D. Wilkins, Pddl–the planning domain definition language, AIPS-98 planning committee 3 (1) (1998) 14.
[32] A. Amos-Binks, J. Clark, K. Weston, M. Winters, K. Harfoush, Efficient attack plan recognition using automated planning, in: Computers and Communications (ISCC), 2017 IEEE Symposium on, IEEE, 2017, pp. 1001–1006.
[33] J. Hoffmann, Simulated penetration testing: From" dijkstra" to" turing test++"., in: ICAPS, 2015, pp. 364–372.
[34] A. Riabov, S. Sohrabi, O. Udrea, O. Hassanzadeh, Efficient high quality plan exploration for network security, in: 11th Scheduling and Planning Applications woRKshop(SPARK), 2016.
[35] M. Backes, J. Hoffmann, R. Künnemann, P. Speicher, M. Steinmetz, Simulated penetration testing and mitigation analysis, arXiv preprint arXiv:1705.05088 (2017).
[36] S.-H. Liao, P.-H. Chu, P.-Y. Hsiao, Data mining techniques and applications–a decade review from 2000 to 2011, Expert systems with applications 39 (12) (2012) 11303–11311.
[37] A. K. Jain, Data clustering: 50 years beyond k-means, Pattern recognition letters 31 (8) (2010) 651–666.
[38] R. Vaarandi, A data clustering algorithm for mining patterns from event logs, in: IP Operations & Management, 2003.(IPOM 2003). 3rd IEEE Workshop on, IEEE, 2003, pp. 119–126.
[39] M. Landauer, M. Wurzenberger, F. Skopik, G. Settanni, P. Filzmoser, Dynamic log file analysis: An unsupervised cluster evolution approach for anomaly detection, computers & security 79 (2018) 94–116.
[40] J. Hipp, U. Güntzer, G. Nakhaeizadeh, Algorithms for association rule mininga general survey and comparison, ACM sigkdd explorations newsletter 2 (1) (2000) 58–64.
[41] Z. Cao, Y. Tian, T.-D. B. Le, D. Lo, Rule-based specification mining leveraging learning to rank, Automated Software Engineering (2018) 1–30.
[42] R. Agrawal, T. Imieliński, A. Swami, Mining association rules between sets of items in large databases, in: Acm sigmod record, Vol. 22, ACM, 1993, pp. 207–216.
[43] R. Agrawal, R. Srikant, et al., Fast algorithms for mining association rules, in: Proc. 20th int. conf. very large data bases, VLDB, Vol. 1215,

1994, pp. 487–499.
[44] A. Gal, Evaluating matching algorithms: the monotonicity principle, in: Semantic Integration Workshop (SI-2003), 2003, p. 167.
[45] S. Parkinson, V. Somaraki, R. Ward, Auditing file system permissions using association rule mining, Expert Systems with Applications 55 (2016) 274 – 283. doi:https://doi.org/10.1016/j.eswa.2016.02.027.
URL http://www.sciencedirect.com/science/article/pii/S0957417416300586
[46] S. L. Morgan, C. Winship, Counterfactuals and causal inference, Cambridge University Press, 2014.
[47] G. E. Box, G. M. Jenkins, G. C. Reinsel, G. M. Ljung, Time series analysis: forecasting and control, John Wiley & Sons, 2015.
[48] C. Bechlivanidis, D. A. Lagnado, Time reordered: Causal perception guides the interpretation of temporal order, Cognition 146 (2016) 58–66.
[49] L. Wang, J. Meng, P. Xu, K. Peng, Mining temporal association rules with frequent itemsets tree, Applied Soft Computing 62 (2018) 817–829.
[50] T.-F. Tan, Q.-G. Wang, T.-H. Phang, X. Li, J. Huang, D. Zhang, Temporal association rule mining, in: International Conference on Intelligent Science and Big Data Engineering, Springer, 2015, pp. 247–257.
[51] Z. Liang, T. Xinming, L. Lin, J. Wenliang, Temporal association rule mining based on t-apriori algorithm and its typical application, in: Proceedings of International Symposium on Spatio-temporal Modeling, Spatial Reasoning, Analysis, Data Mining and Data Fusion, 2005.
[52] E. Winarko, J. F. Roddick, Armada–an algorithm for discovering richer relative temporal association rules from interval-based data, Data & Knowledge Engineering 63 (1) (2007) 76–90.
[53] J. F. Roddick, M. Spiliopoulou, A survey of temporal knowledge discovery paradigms and methods, IEEE Transactions on Knowledge and data engineering 14 (4) (2002) 750–767.
[54] S. Bleisch, M. Duckham, A. Galton, P. Laube, J. Lyon, Mining candidate causal relationships in movement patterns, International Journal of Geographical Information Science 28 (2) (2014) 363–382.
[55] A. Fedorchenko, I. Kotenko, D. El Baz, Correlation of security events based on the analysis of structures of event types, in: Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2017 9th IEEE International Conference on, Vol. 1, IEEE, 2017, pp. 270–276.
[56] E. Nazerfard, P. Rashidi, D. J. Cook, Using association rule mining to discover temporal relations of daily activities, in: International Conference On Smart homes and health Telematics, Springer, 2011, pp. 49–56.
[57] K. Chalak, H. White, Causality, conditional independence, and graphical separation in settable systems, Neural Computation 24 (7) (2012) 1611–1668.
[58] C. Silverstein, S. Brin, R. Motwani, J. Ullman, Scalable techniques for mining causal structures, Data Mining and Knowledge Discovery 4 (2-3) (2000) 163–192.
[59] G. F. Cooper, A simple constraint-based algorithm for efficiently mining observational databases for causal relationships, Data Mining and Knowledge Discovery 1 (2) (1997) 203–224.
[60] S. Mani, G. F. Cooper, Causal discovery using a bayesian local causal discovery algorithm., in: Medinfo, 2004, pp. 731–735.
[61] G. F. Cooper, C. Yoo, Causal discovery from a mixture of experimental and observational data, in: Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc., 1999, pp. 116–125.
[62] M. J. Rattigan, M. E. Maier, D. D. Jensen, Relational blocking for causal discovery., in: AAAI, 2011.
[63] J. Li, T. D. Le, L. Liu, J. Liu, Z. Jin, B. Sun, Mining causal association rules, in: Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on, IEEE, 2013, pp. 114–123.
[64] H. Schünemann, S. Hill, G. Guyatt, E. A. Akl, F. Ahmed, The grade approach and bradford hill's criteria for causation, Journal of Epidemiology & Community Health 65 (5) (2011) 392–395.
[65] A. Sliva, S. N. Reilly, R. Casstevens, J. Chamberlain, Tools for validating causal and predictive claims in social science models, Procedia Manufacturing 3 (2015) 3925–3932.
[66] S. Acharya, B. S. Lee, Incremental causal network construction over event streams, Information Sciences 261 (2014) 32–51.
[67] J. M. Bland, D. G. Altman, Statistics notes: measurement error, Bmj 313 (7059) (1996) 744.
[68] S. Parkinson, M. Vallati, A. Crampton, S. Sohrabi, Graphbad: A general technique for anomaly detection in security information and event management, Concurrency and Computation: Practice and Experience (2018) e4433.
[69] P. R. Wolf, C. D. Ghilani, Adjustment computations: statistics and least squares in surveying and GIS, Wiley-Interscience, 1997.
[70] H. Abdi, Coefficient of variation, Encyclopedia of research design 1 (2010) 169–171.
[71] C. Leys, C. Ley, O. Klein, P. Bernard, L. Licata, Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median, Journal of Experimental Social Psychology 49 (4) (2013) 764–766.
[72] J. Ledolter, Data mining and business analytics with R, John Wiley & Sons, 2013.
[73] G. Piatetsky-Shapiro, Discovery, analysis and presentation of strong rules, Knowledge discovery in databases (1991) 229–238.
[74] K. Garg, D. Kumar, Comparing the performance of frequent pattern mining algorithms, International Journal of Computer Applications 69 (25) (2013).
[75] T. Karthikeyan, N. Ravikumar, A survey on association rule mining, International Journal of Advanced Research in Computer and Communication Engineering 3 (1) (2014) 2278–1021.
[76] H. Ishibuchi, I. Kuwajima, Y. Nojima, Prescreening of candidate rules using association rule mining and pareto-optimality in genetic rule selection, in: Knowledge-Based Intelligent Information and Engineering Systems, Springer, 2007, pp. 509–516.
[77] P. K. D. Sarma, A. K. Mahanta, Reduction of number of association rules with inter itemset distance in transaction databases, International Journal of Database Management Systems 4 (5) (2012) 61.
[78] C. Tew, C. Giraud-Carrier, K. Tanner, S. Burton, Behavior-based clustering and analysis of interestingness measures for association rule mining, Data Mining and Knowledge Discovery 28 (4) (2014) 1004–1045.
[79] S. Vanamala, L. P. Sree, S. D. Bhavani, Rare association rule mining for data stream, in: Computer and Communications Technologies (ICCCT), 2014 International Conference on, IEEE, 2014, pp. 1–6.
[80] D. Hristovski, J. Stare, B. Peterlin, S. Dzeroski, Supporting discovery in medicine by association rule mining in medline and umls, Studies

in health technology and informatics (2) (2001) 1344–1348.
[81] W.-Y. Lin, M.-C. Tseng, Automated support specification for efficient mining of interesting association rules, Journal of Information Science 32 (3) (2006) 238–250.
[82] B. W. Silverman, Density estimation for statistics and data analysis, Routledge, 2018.
[83] S. S. Shapiro, M. B. Wilk, An analysis of variance test for normality (complete samples), Biometrika 52 (3/4) (1965) 591–611.
[84] N. M. Razali, Y. B. Wah, et al., Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests, Journal of statistical modeling and analytics 2 (1) (2011) 21–33.
[85] A. Makarov, G. Simonova, Comparative analysis of the powers of the two-sample kolmogorov–smirnov and anderson–darling tests under various alternatives, Journal of Mathematical Sciences (2018) 1–6.
[86] Y. Xiao, A fast algorithm for two-dimensional kolmogorov–smirnov two sample tests, Computational Statistics & Data Analysis 105 (2017) 53–58.
[87] T. Kirkman, Statistics to use. 1996, URL: http://www. physics. csbsju. edu/stats/Accessed 31 (2007) 12.
[88] A. W. Van der Vaart, Asymptotic statistics, Vol. 3, Cambridge university press, 1998.
[89] A. Makarov, G. Simonova, Some properties of two-sample kolmogorov-smirnov test in the case of contamination of one of the samples., Journal of Mathematical Sciences 220 (6) (2017).
[90] S. Laxman, P. S. Sastry, A survey of temporal data mining, Sadhana 31 (2) (2006) 173–198.
[91] J. Pearl, Causality: models, reasoning, and inference, 2nd Edition, Cambridge University Press, Cambridge, 2009.
[92] J. Pearl, Causality: models, reasoning, and inference, Econometric Theory 19 (675-685) (2003) 46.
[93] D. Heckerman, C. Meek, G. Cooper, A bayesian approach to causal discovery, in: Innovations in Machine Learning, Springer, 2006, pp. 1–28.
[94] K. Yu, J. Li, L. Liu, A review on algorithms for constraint-based causal discovery, arXiv preprint arXiv:1611.03977 (2016).
[95] J. Barnat, L. Brim, P. Rockai, Parallel partial order reduction with topological sort proviso, in: Software Engineering and Formal Methods (SEFM), 2010 8th IEEE International Conference on, IEEE, 2010, pp. 222–231.
[96] P. Spirtes, C. Glymour, An algorithm for fast recovery of sparse causal graphs, Social science computer review 9 (1) (1991) 62–72.
[97] P. Spirtes, C. N. Glymour, R. Scheines, Causation, prediction, and search, Cambridge MA: MIT Press, 2000.
[98] T. Claassen, T. Heskes, A logical characterization of constraint-based causal discovery, arXiv preprint arXiv:1202.3711 (2012).
[99] C. Cheek, H. Zheng, B. R. Hallstrom, R. E. Hughes, Application of a causal discovery algorithm to the analysis of arthroplasty registry data, Biomedical engineering and computational biology 9 (2018) 1179597218756896.
[100] M. Kalisch, P. Bühlmann, Estimating high-dimensional directed acyclic graphs with the pc-algorithm, Journal of Machine Learning Research 8 (Mar) (2007) 613–636.
[101] D. Geiger, T. Verma, J. Pearl, d-separation: From theorems to algorithms, in: Machine Intelligence and Pattern Recognition, Vol. 10, Elsevier, 1990, pp. 139–148.
[102] P. Spirtes, Using d-separation to calculate zero partial correlations in linear models with correlated errors (1996).
[103] T. D. Le, T. Hoang, J. Li, L. Liu, H. Liu, A fast pc algorithm for high dimensional causal discovery with multi-core pcs, arXiv preprint arXiv:1502.02454 (2015).
[104] R. Scheines, D-separation, urlhttps://www.andrew.cmu.edu/user/-scheines/tutor/d-sep.html/, [Online; accessed 07-April-2018] (2018).
[105] D. Entner, P. O. Hoyer, On causal discovery from time series data using fci, Probabilistic graphical models (2010) 121–128.
[106] P. O. Hoyer, S. Shimizu, A. J. Kerminen, M. Palviainen, Estimation of causal effects using linear non-gaussian causal models with hidden variables, International Journal of Approximate Reasoning 49 (2) (2008) 362–378.
[107] J. Zhang, On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias, Artificial Intelligence 172 (16-17) (2008) 1873–1896.
[108] D. Colombo, M. H. Maathuis, Order-independent constraint-based causal structure learning, The Journal of Machine Learning Research 15 (1) (2014) 3741–3782.
[109] J. Tian, Generating markov equivalent maximal ancestral graphs by single edge replacement, arXiv preprint arXiv:1207.1428 (2012).
[110] D. Colombo, M. H. Maathuis, M. Kalisch, T. S. Richardson, Learning high-dimensional directed acyclic graphs with latent and selection variables, The Annals of Statistics (2012) 294–321.
[111] R. A. Ali, T. S. Richardson, P. Spirtes, et al., Markov equivalence for ancestral graphs, The Annals of Statistics 37 (5B) (2009) 2808–2837.
[112] J. Zhang, A characterization of markov equivalence classes for directed acyclic graphs with latent variables, arXiv preprint arXiv:1206.5282 (2012).
[113] J. Zhang, Causal reasoning with ancestral graphs, Journal of Machine Learning Research 9 (Jul) (2008) 1437–1474.
[114] M. Kalisch, M. Mächler, D. Colombo, M. H. Maathuis, P. Bühlmann, et al., Causal inference using graphical models with the r package pcalg, Journal of Statistical Software 47 (11) (2012) 1–26.
[115] S. Shimizu, P. O. Hoyer, A. Hyvärinen, A. Kerminen, A linear non-gaussian acyclic model for causal discovery, Journal of Machine Learning Research 7 (Oct) (2006) 2003–2030.
[116] M. Fox, D. Long, Pddl2.1: An extension to pddl for expressing temporal planning domains, Journal of artificial intelligence research 20 (2003) 61–124.
[117] A. Gerevini, A. Saetti, I. Serina, Planning through stochastic local search and temporal action graphs in lpg, Journal of Artificial Intelligence Research 20 (2003) 239–290.
[118] M. Vallati, L. Chrpa, M. Grzes, T. L. McCluskey, M. Roberts, S. Sanner, The 2014 international planning competition: Progress and trends, AI Magazine 36 (3) (2015) 90–98.
[119] Z. Chen, A. Mukherjee, B. Liu, Aspect extraction with automated prior knowledge learning, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vol. 1, 2014, pp. 347–358.
[120] R. Gargeya, T. Leng, Automated identification of diabetic retinopathy using deep learning, Ophthalmology 124 (7) (2017) 962–969.
[121] F. Salfner, S. Tschirpke, M. Malek, Comprehensive logfiles for autonomic systems, in: 18th International Parallel and Distributed Processing Symposium, 2004. Proceedings., IEEE, 2004, p. 211.

[122] P. He, J. Zhu, P. Xu, Z. Zheng, M. R. Lyu, A directed acyclic graph approach to online log parsing, arXiv preprint arXiv:1806.04356 (2018).