UNIVERSITY OF HUDDERSFIELD

DISCOVERING AND UTILISING EXPERT KNOWLEDGE FROM SECURITY EVENT LOGS

by

SAAD ULLAH KHAN

A thesis submitted in partial fulfilment for the degree of Doctor of Philosophy

in the

School of Computing and Engineering
Centre for Cyber Security

September 2019

Copyright Statement

Abstract

Security assessment and configuration is a methodology of protecting computer systems from malicious entities. It is a continuous process and heavily dependent on human experts, which are widely attributed to 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 determine status of security, such as failures, configuration modifications, system operations 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. Doing so automatically allows for persistent and methodical testing without an excessive amount of manual time and effort, and makes computer security more accessible to non-experts. In this thesis, 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 recommend security improvements.

The proposed solution utilises association and causal rule mining techniques to automatically discover relationships in the event log entries. The relationships are in the form of cause and effect rules that define security-related patterns. These rules and other relevant information are encoded into a PDDL-based domain action model. The domain model and problem instance generated from any vulnerable system can then be used to produce a plan-of-action by employing a state-of-the-art automated planning algorithm. The plan can be exploited by non-professionals to identify the security issues and make improvements. Empirical analysis is subsequently performed on 21 live, real world event log datasets, where the acquired domain model and identified plans are closely examined. The solution's accuracy lies between $73\% - 92\%$ and gained a significant performance boost as compared to the manual approach of identifying event relationships.

The research presented in this thesis is an automation of extracting knowledge from event data steams. The previous research and current industry practices suggest that this knowledge elicitation is performed by human experts. As evident from the empirical analysis, we present a promising line of work that has the capacity to be utilised in commercial settings. This would reduce (or even eliminate) the dire and immediate need for human resources along with contributing towards financial savings.

Acknowledgements

First of all, I would like to express my deepest gratitude to Dr. Simon Parkinson. Without his mentorship, patience, commitment, determination and motivation, I would have never been able to finish my Ph.D. I learned a lot of skills under his supervision, but there is something that I want to bring in the spotlight; he taught me what it really means to critically understand and analyse any given problem, and perform a complete, thorough and frenetic research, the one that does not let you sleep at night. Throughout my Ph.D. journey, Dr. Simon Parkinson provided an excellent, friendly and encouraging research atmosphere along with the very best experience and valuable insights.

Many thanks to the University of Huddersfield for awarding me the Vice Chancellor scholarship. I am very grateful for their financial support and generosity. The scholarship allowed me to keep my entire focus on the research work.

Additionally, I would to thank and acknowledge all members of staff and school administration, who always provided a quick and constant support in resolving all of my Ph.D. related issues.

Contents

List of Figures

List of Tables

Abbreviations

ADL	**A**ction **D**escription **L**anguage
AI	**A**rtificial **I**ntelligence
AR	**A**ssociation **R**ule
ARM	**A**ssociation **R**ule **M**ining
AP	**A**utomated **P**lanning
ASCol	**A**utomated **S**tatic **Co**nstraint **L**earner
BLCD	**B**ayesian **L**ocal **C**ausal **D**iscovery
BFS	**B**readth-**F**irst **S**earch
CBR	**C**ase-**B**ased **R**easoning
CI	**C**onditional **I**ndependence
DFS	**D**epth-**F**irst **S**earch
DAG	**D**irected **A**cyclic **G**raph
EDF	**E**mpirical **D**istribution **F**unction
EUROPA	**E**xtensible **U**niversal **R**emote **O**perations **P**lanning **A**rchitecture
FCI	**F**ast **C**ausal **I**nference
FF	**F**ast-**F**orward
FSM	**F**inite **S**tate **M**achines
FD	**F**requency **D**istribution
GPS	**G**eneral **P**roblem **S**olver
GIPO	**G**raphical **I**nterface for **P**lanning with **O**bjects
ICKEPS	**I**nternational **C**ompetition on **K**nowledge **E**ngineering for **P**lanning and **S**cheduling
(ISC)^2	**I**nternational **I**nformation **S**ystem **S**ecurity **C**ertification **C**onsortium
IPC	**I**nternational **P**lanning **C**ompetition

KA	**K**nowledge **A**cquisition
KR	**K**nowledge **R**epresentation
LOCM	**L**earning **O**bject-**C**entric **M**odels
LHS	**L**eft-**H**and **S**ide
LiNGAM	**Li**near **N**on-**G**aussian **A**cyclic **M**odel
LCD	**L**ocal **C**ausal **D**iscovery
LPG	**L**ocal search for **P**lanning **G**raphs
LoCI	**L**ogical **C**ausal **I**nference
MI	**M**arginal **I**ndependence
MB	**M**arkov **B**lanket
MBCS	**M**arkov **B**lanket/**C**ollider **S**et
MAGs	**M**aximal **A**ncestral **G**raphs
MISL	**M**ulti-level **I**ntensive **S**ubset **L**earning
NASA	**N**ational **A**eronautics and **S**pace **A**dministration
NOAH	**N**ets of **A**ctions **H**ierarchies
NDDL	**N**ew **D**omain **D**efinition **L**anguage
OFD	**O**bject **F**requency **D**istribution
OS	**O**perating **S**ystem
PAG	**P**artial **A**ncestral **G**raph
POMDP	**P**artially **O**bservable **M**arkov **D**ecision **P**rocesses
PC	**P**eter-**C**lark algorithm
PDDL	**P**lanning **D**omain **D**efinition **L**anguage
PDS	**P**ossible-**D**-**S**EP or **P**ossible **D**-**S**eparation
PPDDL	**P**robabilistic **PDDL**
RND	**R**eference **N**ormal **D**istribution
RHS	**R**ight-**H**and **S**ide
SID	**S**ecurity **ID**entifier
SIEM	**S**ecurity **I**ncident and **E**vent **M**anagement
SW	**S**hapiro–**W**ilk
SEC	**S**imple **E**vent **C**orrelator
SLAF	**S**imultaneous **L**earning and **F**iltering
SD	**S**tandard **D**eviation
STRIPS	**S**tanford **R**esearch **I**nstitute **P**roblem **S**olver

SLPR	**S**tructure **l**earning using **P**rior **R**esults
SR	**S**upport **R**ange
TA	**T**emporal-**A**ssociation
TAA	**T**emporal-**A**ssociation **A**ccuracy
TAC	**T**emporal-**A**ssociation-**C**ausal
TSKS	**T**wo-**S**ample **K**olmogorov–**S**mirnov
USM	**U**nified **S**ecurity **M**anagement
V&V	**V**alidation and **V**erification
VTEC	**V**ariable **T**emporal **E**vent **C**orrelator
WFP	**W**indows **F**iltering **P**latform

Symbols

Symbol	Meaning
μ	Mean
s	Standard Deviation
$\rightarrow, \leftarrow$	Implies
$\subseteq$	Subset
$\neq$	Not equal to
$-$	Undirected Edge
C	Set of temporal-association rules
$\emptyset$	Null or empty set
$\forall$	For every element in a set
$\in$	Belongs to
$\notin$	Does not belong to
$<$	Less than
$>$	Greater than
$=$	Equals to
$\neq$	Not equal to
$\geq$	Greater than or equal to
$\leq$	Less than or equal to
$\perp$	Conditionally independent of
$\mid$	Given this condition
ϕ	Path of one or more connected vertices
a, A	Action, Set of actions
s, S	state, Set of State
γ	State-transition functions
$\sum = (S, A, \gamma)$	State-transition system

f	Numeric fluents
i	Intial state
g	Goal state
$P = (s, f, a, i, g)$	A planning system

Chapter 1

Introduction

This thesis work concerns the area of learning domain action models through system event logs for Artificial Intelligence planning systems. This chapter provides the overview, research question, motivation, rationale, novel contributions of the thesis and a reader's guide to the thesis structure.

1.1 Overview

Many organisations are vulnerable to critical security threats exposed due to their digital infrastructure, and given the continuously increasing size and nature of their business operations, there is a need to pro-actively identify and mitigate security vulnerabilities. This process requires expert and up to date knowledge of the security threats and how they can be eliminated. Such knowledge is in short supply, costly, sometimes unavailable and requires a high amount of human effort [2]. Businesses are facing challenges with recruiting and maintaining cyber-security expertise within their organisation and as a result, they are often unable to adequately secure their systems. According to a survey[1] conducted in 2018 by International Information System Security Certification Consortium, or $(ISC)^2$, 2.93 million cybersecurity positions remain open and unfilled around the world, depicting the lack of skilled security professionals. Despite the skills shortage in cyber analysis, it is imperative to maintain the system security against malicious attacks.

[1] https://www.isc2.org/Research/Workforce-Study

The lack or absence of expertise can be resolved using computational intelligence to autonomously perform security evaluation of a machine and generate mitigation plans. There is potential to develop a system whereby expert knowledge is automatically captured through monitoring experts' system interactions and therefore minimises any overhead of knowledge creation as it can be captured during or after live analysis/configuration tasks. Non-expert users will be able to obtain the desired knowledge. This enables the use of expert knowledge, either in an assistive mode for training purposes or autonomous mode to execute the knowledge model to determine if a vulnerability exists as well as recommend automated configuration activities. Besides security, researchers from various other domains are also exploring automation techniques, such as in inferring behaviour models from execution traces of object-oriented programs [3], mining specifications directly from software libraries [4], merging existing specification miners to produce a superior specification [5] and automatically discovering program flaws by learning method calls [6].

This research work presents an automated solution that can extract security actions performed on a system using event log entries. An event log dataset is a discrete, ordered sequence of event instances that contains information regarding the activities of the system; generated by either a user, application or the operating systems (OS) itself [7]. Event logs are used as a tool for tracking user, application and system changes (e.g. creating or deleting users), along with troubleshooting the system failures and other related issues. The proposed solution assembles the knowledge by discovering complex chains of event relationships that represent linked human expert activities. Therefore, every chain of connected events is the representative of a specific security-related action. In this thesis, the relationships are termed as 'temporal-association-causal' rules, which are obtained by incorporating a notion of time to frequently co-occurred events and subsequently applying causal inference to prioritise the discovered event relationships. The rules are then used by the automated planning tools to provide appropriate knowledge to the non-expert users. The proposed solution is capable of providing a technological solution to help reduce the security skills shortfall, enabling expert-like decision making capabilities for users without cyber security expertise. It is reasonable to suggest that a large amount of users are interacting with the vulnerable machines, and providing an automated solution with access to skilled expertise has potential to have a positive impact on overall cyber resilience.

The developed technique is currently tested on Microsoft Window OS [8], but the approach is independent of OS and can easily be applied to the event logs of other systems or sources. The data recorded by Microsoft event logging mechanism is fully-structured and represented in the form of object-value pairs. A default set of classes exists in the .NET framework for identifying and parsing all categories of events, which facilitates an easier development and creation of a valid testing environment for the proposed approach. The future extensions of this research will extract features focusing on other, widely used Linux- and Mac-based systems along with web-servers, OpenStack cloud [9] and Fog computing platforms [10].

1.2 What Are Event logs?

Event logs are a powerful source of knowledge that can be used for understanding operations, performance and security status of the underlying system [11]. Event logs are often described as the gold mine of knowledge due to their extensive and detailed record-keeping ability. According to Microsoft documentation, the events are divided into five different levels: Information, Warning, Error, Success Audit, and Failure Audit and categorised into separate Application, Security, Setup, System and Forwarded logs. A complete explanation of the general types of event logs, along with the classification of security events, is provided in Appendix A. This research work only focuses on the security events, which are easily identifiable within the existing entries. The security events contain structured data that can be easily extracted and processed by a software application to provide detailed state information on a system's security controls. These events include records of failed login attempts, system operations based on the firewall policy, changes in user or application permissions, cryptographic operations and many other security-related activities. The system activity is recorded based on the security auditing and configuration policy, which is predefined by the administrator. Each entry is created from their prototype/blueprint, called event type. Any event type can have many instances or entries, and each entry consists of a series of objects that define a particular occurrence of an event. The objects include a unique identifier, user name, event source, machine name, time-stamp, and application and service-specific objects.

The security events of a system provide a detailed record of the security controls, which describes both unauthorised activities and configuration events [12]. Many techniques

have been developed in previous research to identify relationships among events. Such techniques generally employ variations of clustering [13], correlation rule mining [14] and causal inference [15] algorithms. Most of the existing solutions construct a security model that determines the causes behind system failures and suggest or in some cases implement remedial actions. Although these techniques are focused on identifying problems using software agents, there is a great potential to adopt this same philosophy for identifying missing or incomplete security configurations as well as the security issues. It should be noticed here that a configuration process against a certain issue usually generates a large number of events, according to the specified audit policy, and produces a complex chain of events to record the whole activity. The audit policy of a system defines the types of events stored in the Security log. The Microsoft OS uses nine audit policy categories and fifty subcategories for fine-grained control over which the information is logged. To the best of our knowledge, there are no autonomous Security Incident and Event Management (SIEM) solutions aimed at capturing actionable knowledge from the event log sources that can be exploited by an intelligent software agent to aid non-expert users.

It would be of great benefit to automatically determine and extract chains of events against configuration activities. As an example, consider a scenario where a system administrator is maintaining a file server. An unauthorised user attempts to view a file named $confidential.doc$ and successfully gains access to it. To document this activity, an event of type 4663 will be recorded along with the relevant objects that shows 'An attempt was made to access an object'. Assuming the administrator takes notice of this event and realises that the user is not authorised to access any file on this server and their permissions should be immediately removed. The administrator proceeds to alter the network share rights and permissions. As a result, an event log entry will be logged with the type of 4670, detailing that 'Permissions on an object were changed'. The event logging mechanism generates this event in a way where it keeps a record of the new and previous set of permissions. Many additional event log entries might also be generated during the time span of events 4663 and 4670 due to other system activities, and therefore it would require expert knowledge, time and effort to establish the connection between events. Despite being simple, this example describes the fact that managing security requires both experience and expert knowledge. A system that can discover a relationship between events 4663 and 4670 in an automated manner would

be beneficial for the non-expert users. It would enable the immediate investigation of such event entries, and determine if any user is exercising more than the required set of permissions.

1.3 Assumptions

Following are the assumptions made in this thesis:

1. **The event log was not cleared at any point in time**. All event logging mechanisms allow the user to clear or remove the entries of an event log. So, if a user clears the event log, it would delete one or more entries. Processing such event log entries by the proposed solution would output an incomplete set of event relationships. This can be detrimental for the non-expert users in terms of identifying all security issues and making improvements;

2. **The event log dataset was retrieved directly from the system, not through any intermediary tool.** For extracting knowledge, the event log dataset should be in raw format and acquired directly from the main source. Involving third-party tools to obtain any dataset might alter the event entries and lead towards erroneous results. Moreover, all event logs should be acquired and formatted through the same mechanism for accurate results; and

3. **The user, application and system activities stored in the event logs have satisfactory coverage of the required security knowledge.** Any machine that is being used to extract the expert security knowledge needs to be properly configured/secured. The machine's event log should possess a set of entries that are required to be generated to document a certain security-related activity. It should be noticed here that according to the default audit policy, all categories and sub-categories are not enabled. Therefore, it is possible that the event log does not have a full record of performed activities, and the proposed solution produces invalid event relationships; and

4. **The machine from which the knowledge is acquired and the machine on which the knowledge is applied have similar settings.** The configuration, environment and intended use of the vulnerable machine are the same as of the

machine from which the domain action model was extracted. The similar operating system, settings, and context of events are required to enable a compatibility between the learned security knowledge and its utilisation for making the necessary changes.

1.4 Research Question

The research question addressed in this thesis is whether it is possible to automatically extract security-related actions (i.e. knowledge) performed by an expert on a machine, and allow non-experts to utilise the actions for auditing and improving the security of their systems. The current knowledge mining techniques rely on human experts, where the knowledge is collected and modelled either manually or via assistive tools. Exploring this question led to the development of an unsupervised knowledge acquisition technique, which can discover (temporally ordered) cause and effect relationships among the event log entries of any given machine. The terms 'machine' or 'system' used in this thesis refers to any computing device, such as personal computer, laptop, server, etc.

This thesis presents the development and evaluation of a generalised approach, which can be applied to any event log dataset that is or can be structured in a certain format. The approach employs correlation and causal mining algorithms due to their known suitability in finding such relationships. 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. Therefore, the proposed approach allows non-expert users to conduct expert actions without the presence of a human expert. Further to that, the automated solution can perform frequent monitoring, analysis and mitigation activities. By doing this, potential security threats and vulnerabilities can be identified quickly and mitigation activities can be put in place.

1.5 Scope, Motivation And Aim

Aim: To provide an automated method of extracting security knowledge by learning the expert user activities performed on a machine, hence reducing the amount of required knowledge, resources and effort required by non-expert users to identify and eliminate security issues.

The non-expert users do not possess sufficient knowledge and/or experience to recognise and comprehend the security issues. The work presented in this thesis provides a method that can automatically enable all kinds of users to perform the security analysis without needing expert knowledge, along with suggesting remedial actions. The primary driver behind this research is to reduce the disparity in system security between those configured by experts and non-experts. This will ensure a higher level of security is maintained for all users. The automated process of building security knowledge base will further reduce the manual effort and allow better efficiency in terms of resources and time. Another motivation of this research is to create a solution that is generic and universally applicable.

The scope of this work is to explore how, and to what extent, knowledge can be acquired automatically from the existing data sources. Moreover, how that knowledge can be utilised by automated planning algorithms for the intelligent application and optimisation of the security issues identification and elimination process. The focus is on obtaining useful knowledge that can strengthen the security of any given machine by mimicking the traditional human expert analysis. The philosophy of this thesis is well-justified because it automates and enhances normal industry practice. This work can be viewed as a step towards an increase in autonomous knowledge elicitation and utilisation in the domain of cybersecurity. The thesis will also examine the performance of the proposed solution in terms of time, memory and processing power consumption to demonstrate its application in the real-world environment.

1.6 Objectives

The objectives with respect to the security knowledge mining are:

1. The first objective is to develop a mechanism that can gather and process the user activities data into a single, structured format; and

2. The second objective is to build a technique for automated knowledge extraction that can utilise the formatted data and identify the relationships among the user activities.

The objectives concerning the automated planning are:

1. The first objective is to produce a domain action model of the extracted relationships in a standardised language. This will allow state-of-the-art, domain-independent automated planning tools to generate both accurate and temporally ordered optimal plans;

2. The second objective is to convert the automated plan into a format that can be easily understood by the non-expert users; and

3. The third objective is to acquire data from live, real-world data for evaluation and demonstrate the accuracy, performance, and usability of the proposed solution.

1.7 Contributions

To the best of the author's knowledge, this is the first work in the domain of Security Incident and Event Management (SIEM) systems. It presents a mechanism for automated knowledge acquisition from event log sources, along with an intelligent utilisation of the elicited knowledge. The novelty of this research lies in the automated learning, modelling, and planning of security event relationships by applying correlation and causal rule mining coupled with the generalised automated planning.

The summary of the primary thesis contributions to the subject area is as follows:

1. **A novel technique for extracting and representing security knowledge from event log entries**. This technique starts by creating an *object-based* model of event log entries, which is processed by an association rule mining algorithm. The association rules are then converted into sequences of event relationships using a temporal metric, which are further validated by applying a causal inference

technique. The extracted knowledge is represented in the form of a domain action model. The problem instance is generated from a previously unseen computer that requires security improvements. Together with the domain model and problem instance, plan solution is generated by the automated planning algorithm and utilised by the non-expert users for configurations;

2. **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 without any human input used by the association rule mining algorithm. The values are derived from the object-based model of the 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;

3. **Software implementation**. A novel software tool that is capable of learning knowledge from mining Microsoft event log sources. The software represents the output of using acquired knowledge on previously unseen machines in a user-friendly manner. Several other tools have also been developed that can serve the non-expert user for debugging purposes; and

4. **Empirical analysis**. The developed tool is evaluated on event log datasets taken from live, multi-user and network-based machines to determine the overall performance and accuracy of the proposed technique. This empirical analysis also demonstrates and objectifies features of the proposed solution from various aspects, such as usability, efficiency, and productivity.

The solution presented in this thesis is inspired by on-going research in applying Artificial Intelligence techniques in the area of cybersecurity. It is expected that the contributions made in this research will entice more researchers towards the subject.

1.8 Publications

The following publications are related to this PhD project:

1.8.1 Journal Papers

Khan, S., Parkinson, S. (2018). Eliciting and utilising knowledge for security event log analysis: An association rule mining and automated planning approach. Expert Systems with Applications, 113, 116–127.

Parkinson, S., Khan, S. (2018). Identifying irregularities in security event logs through an object-based Chi-squared test of independence. Journal of information security and applications, 40, 52–62.

Khan, S., Parkinson, S. (2019). Discovering and utilising expert knowledge from security event logs. Journal of Information Security and Applications, 48, 102375.

1.8.2 Conference Papers

Khan, S., Parkinson, S. (2017). Causal Connections Mining Within Security Event Logs. In Proceedings of the Knowledge Capture Conference (p. 38). ACM.

Khan, S., and Parkinson, S. (2017). Towards Automated Vulnerability Assessment. In: 11th Scheduling and Planning Applications woRKshop (SPARK), 19th June 2017, Carnegie Mellon University, Pittsburgh, USA.

Saad, K., Simon, P. (2017). Towards a multi-tiered knowledge-based system for autonomous cloud security auditing. In: Proceedings of the AAAI-17 Workshop on Artificial Intelligence for Cyber Security (pp. 187–194). AAAI

1.8.3 Book Chapters

Khan, S., Parkinson, S. (2018). Automated Planning of Administrative Tasks Using Historic Events: A File System Case Study. In Guide to Vulnerability Analysis for Computer Networks and Systems (pp. 159–182). Springer, Cham.

Khan, S., Parkinson, S. (2018). Review into State of the Art of Vulnerability Assessment using Artificial Intelligence. In Guide to Vulnerability Analysis for Computer Networks and Systems (pp. 3–32). Springer, Cham.

1.9 Thesis Structure

The document is organised as follows:

Chapter 2 reviews and analyses existing domain learning techniques, automated planning, correlation mining and causal inference techniques, alongside the critical literature analysis to highlight the importance of the proposed system.

Chapter 3 describes the process of rule mining from event log dataset. It starts by detailing and motivating the pre-processing phase of event entries and elaborates the process of generating object-based model. Following on, it explains the association rule mining process using an automatically calculated support value. Here, the association rules are represented by their respective event types, where any sequence of events are related through their objects. Next, it presents the process of filtering insignificant rules based on a temporal metric. The remaining individual rules are combined to form sequences of events, where each sequence represents a single configuration activity. Finally, it 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.

Chapter 4 starts by formalising the classical planning and knowledge modelling techniques. Following on, it presents a process of encoding the extracted event relationships into a domain action model, which leads onto the explanation of the automated method for extracting the problem instance from vulnerable machines, and applying an automated planning technique to generate plan-of-action for security improvements.

Chapter 5 is divided into two parts. The first part presents the implementation details of knowledge acquisition and representation applications. The second part presents the evaluation methodology and results in terms of performance and accuracy using the security event logs of live machines.

Finally, Chapter 6 concludes the thesis and provides the future plans and extensions of this research work.

Chapter 2

Background and Literature Review

The purpose of this chapter is to survey and review the existing knowledge acquisition technologies that are relevant for this thesis. It starts by exploring the available data sources that can provide the required security knowledge. Next, it analyses the existing manual, assistive and automated approaches that can extract security knowledge from the data sources and distribute it to non-expert users. In the end, it performs a brief analysis to demonstrate the current state and limitations of the knowledge acquisition systems.

2.1 Data Sources

Many resources are readily available for individuals, who are seeking to learn knowledge and gain expertise in a security-related area. Knowledge can be defined as the theoretical or practical understanding of a phenomenon or an area. Knowledge can be of different types and is presented in the form of postulations, facts, concepts, principles, procedures, models, cognition, heuristics and examples. The resources for presenting security knowledge include books, journals, articles, online discussion forums, vulnerability and solution databases, tutorials, hands-on training sessions, certifications, human experts and many others [16]. Knowledge sharing is a process of exploiting existing knowledge by identifying, transferring and its application to solve tasks in a better, faster and cheaper

manner [17]. A major issue with such type of knowledge sharing is the requirement of effort and time to probe the resources and manually learn about security issues and remedial operations. Another limitation is that the user needs to know what skills they are lacking to perform the security assessment and configuration of a specific system. Moreover, due to a large amount of diversity in information systems, it is difficult for novice users to appropriately secure all aspects of a system.

In this research, we selected event logs as a source of security knowledge. The event logs are often referred as a 'gold mine' of information because they possess a record of all activities performed on the machine. The knowledge in the event logs is acquired through several pattern analysis techniques. The patterns are represented by a set of rules, where each rule contains two or more related events depicting a particular security-related activity. The rules can be formulated using a manual, assistive, or automated approach. All of these approaches are explained in the remaining sections along with their advantages and disadvantages.

2.2 What Is Knowledge Acquisition?

Knowledge Acquisition (KA) is a process of extracting and encoding the knowledge from a certain data source into a form that can be utilised by knowledge-based systems [18]. This process generally consists of several phases: identifying the source of knowledge, developing and applying a technique to extract the knowledge, representing the knowledge in a way that can be understood by a software agent, applying the knowledge to solve given problems along with the provision of explanations and justifications for the steps taken, and a continuous cycle of updating and maintaining the knowledge.

The main purpose of KA is to build a system that can provide expert knowledge regarding a certain subject. The KA can be a long, tiring and expensive process as it is difficult to capture accurate knowledge. The KA is recognised as the major bottleneck for knowledge-based systems, limiting the wider adoptions of such systems [19]. The main difficulty lies in the process of extracting expert knowledge with reasonable accuracy, efficiency, and robustness. This is the reason several KA techniques have been developed over the years. Such techniques can be taxonomised into manual, assistive

and automated approaches, and have their own advantages and disadvantages. These techniques are described briefly in the following sections.

2.3 Manual And Assistive Knowledge Acquisition

The manual and assistive techniques of extracting knowledge are based on the help and underpinnings of human security experts, who manually extract the knowledge from the data sources. The process of knowledge discovery is either fully manual or semi-manual using some supporting tools. The acquired knowledge is then used to automatically identify vulnerabilities of a system and alert the user to implement corrective solution. Several methodologies are available for this approach, and some of them are explained in the following:

2.3.1 Rule-Based Systems

One of the most common method of acquiring and storing knowledge is the rule-based systems. In rule-based systems, human experts use their knowledge and experience to define patterns among events using condition-action relationships (rules). Many tools and techniques are available that can be used for creating, storing and applying rules. A patented research work developed a multi-tier security incident response system [20], which is based on user-defined and configurable rules. The rules are created as open-ended expressions, which can be both conjunctive and disjunctive in operation. The purpose of this system is to identify critical threats and complex attack patterns, notify the user and then generate human-readable reports.

A tool, named Simple Event Correlator (SEC), was developed for network and security management. The SEC is an open-source and platform-independent tool[1]. The tool has various significant features and uses a predefined knowledge base of hard-coded, static rules to monitor and detect event patterns [21]. The rules are created as IF–THEN statements to represent the condition of a specific security issue and propose administrative action to resolve it. Another tool, called Logsurfer [22], conducts the patterns and relationships detection among the events of a given log file. The rules are defined with the help of manually created regular expressions. An important feature of the

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

Logsurfer is the ability to encode context of an event in the rules. The context describes all relevant data for diagnosing a problem and keeps track of time, users, configuration, environment etc. Another similar tool, called Simple Log Watcher[2] (previously known as Swatch), uses regular expression based rules to identify event patterns and perform specified actions. Both regular expressions and corresponding actions are written by the user in C programming language.

Another interesting tool, named Variable Temporal Event Correlator (VTEC) [23], allows the user to define, modify and utilise event relationship rules. The VTEC uses Perl language to define condition-action rules, which are directly applicable on computer networks as well. An important feature of VTEC is efficiency; it is capable of applying rules on large amounts of event data using a multi-core, distributed architecture. Another tool, called Prelude[3], is a rule-based event correlation engine, where the rules are written manually using Lua programming language. Each rule is a function that has a particular input and output alert. This tool can also perform log analysis and general event management operations. An open-source tool, called AlienVault OSSIM[4], performs the event collection, normalisation and correlation. The main achievement of OSSIM is the modelling of cyber-attacks using event correlation rules. It also assigns probability and priority values to an asset for providing better risk management and security visibility. The rules are written manually using XML-like directives, and can also enforce the security policies of a company.

Besides these security applications, rule-based systems are used in several other areas. For example, a research work proposed a rule-based approach to determine energy efficiency anomalies in smart buildings [24]. The rules were developed with the help of human experts and data analysis techniques. After testing on a real-time database, the authors claim that the developed system successfully detected the anomalies along with quantifying the energy consumption behaviour. Another interesting study proposed a rule-based solution to create animations of Java algorithms [25]. The rules are manually written using a tool called, Constraint Handling Rules. Given an algorithm, the solution starts by identifying the interesting segments of the algorithm, and then dynamically creates visual objects and effect to produce animations. Apart from the applications and advantages, one of the main problem associated with the existing rule-based systems is

[2]https://sourceforge.net/projects/swatch/
[3]https://www.prelude-siem.org/projects/prelude/wiki/PreludeCorrelator
[4]https://www.alienvault.com/products/ossim

that it demands domain knowledge, skill and manual effort for extracting and defining complex patterns. Furthermore, the rules production can be a challenging task for larger and dynamic systems as it requires constant maintenance and management to encode new knowledge and updating the existing ones.

2.3.2 Case-Based Reasoning Systems

The case-based reasoning (CBR) approach mainly consists of two parts: case repository and reasoning engine. The case repository is a collection of past cases, containing the knowledge of security issues and their corresponding solutions. The initial repository is developed by human experts. The inference engine takes the (unfamiliar) problem, identify the most similar case in the repository and retrieve the best-suited solution. If the solution is found to be correct, the given problem is formulated into a new case and stored in the repository. A research study [26] acquired the knowledge for case repository by processing large amounts of security content in the textual format. They used text-analytic and clustering algorithms to sanitise and categorise the text and used that to manually build the knowledge for case repository. After that, they employed CBR to analyse event logs and discover the patterns within system maintenance activities to reduce the number of crashes and prevent failures. Another research study applied CBR approach on temporal log data for intelligent monitoring and diagnosis of workflows of a software system [27]. The knowledge and context for case repository was captured with the help of software users. The authors concluded that the solution found and classified given workflow-related problems with high accuracy and efficiency.

Another study presented an assistive solution for security and forensic investigators that can profile attackers using CBR [28]. The experiments conducted has shown positive results in exposing the identity of humans behind the attacks as well as determining if the attack was successful. A similar study built a report generation mechanism using CBR that can save security analyst time [29]. This solution can automatically fill the corresponding templates of known threats and policy violations on the local network. According to the authors, this solution can manage the slight deviations in attacks; however, it cannot handle new, zero-day attacks. One of the major disadvantages of CBR is its inability to represent problems and include the notion of time that is crucial for defining the ordering of security-related actions [30]. Without conclusive ordering,

the CBR cannot produce quality solutions. Furthermore, building a case repository that includes all possible security threats and their solutions is not feasible for practical application as it would require a substantial amount of time and effort.

2.3.3 Codebook-Based Systems

The codebook-based systems define the event relationships by using coding techniques [31]. In this approach, the human expert performs the localisation process, where connections between observable behaviour of the system and actual underlying problem are analysed to identify the set of responsible events, i.e. codebook or knowledge. Every connection made is assigned a codeword. After that, for every possible problem, a binary vector is created to determine if each event in the codebook can be triggered by that particular problem. The elements in a vector can either be one, to depict a relationship between the event and problem, or zero if there is no relation. The codebook (or set of events) is required to be sufficiently large so that it can cover all possible problems of a system. For the problem identification process, events of the underlying system are monitored in real-time and formed into vectors. These event vectors are compared with the existing codebook event vectors to find the most similar match and notify the user with respective problem.

Another research study proposed a codebook-based approach to enhance the performance, efficiency and usability of an anomaly intrusion detection (AID) system [32]. Initially, the normal/legitimate network traffic profiles were used to train a large structural codebook with 541,048 data items, each having a unique index. To find network intrusions or deviations from the codebook, vector quantisation framework was employed that determines similarity between feature vectors, along with reducing vector dimensions. The results have demonstrated high detection rate, lower computational cost and real-time intrusion detection. Apart from requiring a significant amount of time and effort in building and indexing a codebook, these systems are not very productive for dynamic environments, where a slight change in system configuration can cause an ambiguity in matching with the corresponding codeword.

2.3.4 Voting-Based Approach

The voting approach is based on problem-opinion architecture, where each expert provides an opinion against a given problem and the absolute majority opinion is used as a solution [33]. Usually, the opinion does not provide an exact solution but rather a general direction to solve the problem. A research study employed the neural networks to implement voting approach for detecting poison messages in large-scale networks [34]. The poison message exploits a software or protocol vulnerability by sending a malicious message in the network and cause certain machines to fail. The neural network was trained with the help of human experts, using various kinds of poison messages along with opinion weights and biases. The evaluation of this solution shows the accuracy range from $55\% - 75\%$. Another study proposed a voting-based approach for an efficient detection of unknown malware [35]. Initially, datasets were prepared by extracting and using several features of executable files and trained an ensemble of classification algorithms, namely K-nearest neighbours, C4.5 and Ripper. The authors applied three different voting-based approaches to create the ensemble; majority voting, veto voting and trust-based veto voting. The ensemble consolidates the individual decision of each classifier in a way that can classify new malware samples, hence improving the decision strategy, and consequently increases the overall detection capability of the solution. It should be noticed here that most of the voting-based approaches require extensive input from the human operator, which is why they are usually deemed unsuitable for large-scale and complex systems.

2.3.5 Commercial Event Correlation Software

There are several commercial solutions that enable the development and elicitation of event correlation rules. These solutions fall under the category of Security Incident and Event Management (SIEM). Majority of the rules are acquired manually through experts and community of users. One such solution, called Unified Security Management (USM)[5], is developed by AlienVault. The main purpose of this solution is to collect and analyse the log data from multiple applications and devices, and identify malicious patterns related to data security. These tasks are performed using event correlation rules, which are constantly being developed by the company's internal experts. Another

[5]https://www.alienvault.com/products/usm-anywhere

solution is developed by Logrythm[6] that combines correlation rules with Artificial Intelligence to identify malicious behaviour of users and applications. The main benefit of this solution is the context learning of events, detection of statistical anomalies and ability to assign risk level to an asset. The correlation rules are provided by the company and can be developed by the users as well.

2.4 Automated Knowledge Acquisition Approaches

In order to discover relationships or patterns of events for knowledge discovery, a complete set of information is required about the structure, properties, triggers and dependencies of a system. Such information can be acquired automatically, or manually from human experts. It is obvious that the second method requires a large amount of effort, time, resources and experience. The manual techniques are only feasible if the majority of supplied knowledge remains constant over an extended period, i.e. static knowledge. Unlike these conventional systems, if the knowledge is mostly dynamic (changes or requires updating over time), a self-learning, automated approach would be more suitable [36]. Although this approach does not require human assistance in generating and maintaining knowledge, there is a possibility of acquiring unstable and incorrect knowledge if not implemented properly. Recent progress in the research has shown promising techniques in terms of accuracy, productivity and efficiency. Some of the automated techniques are discussed in the following:

2.4.1 Clustering Techniques

A variety of classification algorithms exist that are based on either supervised or unsupervised learning. These algorithms are commonly utilised for knowledge acquisition by exploring the relationships among data items. The unsupervised classification is termed as clustering or exploratory data analysis. Data clustering is a process of grouping items or entries of a finite unlabelled dataset, such that the similar entries with respect to features and attributes are classed together in the same clusters [37]. This process creates one or more segregated groups of entries, where each entry is more related to the other entries of the same group than those of the other groups.

[6] https://logrhythm.com/use-cases/advanced-correlation/

The clustering algorithms can be broadly categorised into Partitional and Hierarchical schemes [38]. The Partitional clustering schemes are parametric, where the algorithms, such as K-means, divide the data into a predefined number of clusters specified by the user. The Hierarchical clustering schemes are non-parametric, where the algorithms, such as MeanShift, decide the number of clusters to create based on certain metrics. The clustering of data items is performed based on a 'similarity' metric [39]. The K-means algorithm groups the data points based on closeness to the centroid of clusters [40]. The closeness is measured with one of the following methods: Euclidean distance, Manhattan distance, Mahalanobis distance and so on. This algorithm starts by taking the desired number of clusters, and randomly assigns each data point into a certain cluster. Then it computes the centroid of each cluster and re-assigns the data points to the closest centroid. This step is repeated until each centroid position becomes stationary. At this point, if the number of iterations are not defined, it will terminate the clustering process, otherwise, it will continue to execute without changing anything. The MeanShift algorithm [41] starts by selecting a random data point and creates a radius of specific length around it, encompassing other data points. The radius length can be defined by the user or automatically determined by the algorithm. The algorithm calculates the centre of all data points (one after another) that are present within the same radius. This process is continued until the centre stops shifting, and a cluster is formed. The algorithm iterates over the remaining data points to find all clusters. Another important distinction in the clustering algorithms is the hard and soft cluster assignment [42]. In hard clustering, each entry is classified into exactly one cluster, whereas in soft clustering, one data entry can be a member of multiple clusters.

A recent paper has proposed a clustering technique to evaluate the Distributed Denial of Service attack security situation [43]. It starts by finding the risk indexes of all machines in a network, merges them and creates a fusion feature set. After that, Fuzzy C-means clustering algorithm is employed to categorise machines into one of the five security classes to determine the level of threat. The real-time experiments have shown this technique to be reasonable and effective. One of the main applications of clustering algorithm is in detecting anomalies and intrusions in a network. A research study proposed a combination of hierarchical clustering and support vector machines approach to detect intrusions in a network with 95.75% accuracy [44]. The original feature set is first processed for quality improvement and size reduction. By training on the new dataset,

experimental results of the new technique have shown higher performance, especially for the denial of service and probing attacks. Clustering techniques have also been utilised for log analysis. For example, a study presented a novel clustering algorithm, called the Simple Logfile Clustering Tool [45], which identifies infrequent events based on outliers that could represent previously unknown fault conditions, irregularities, and other anomalies. Although clustering techniques have been used in many real-world applications, they are not suitable for identifying knowledge in our research, given the nature of event logs data and required output.

A major drawback of using clustering to learn relationships among events is that it may find a connection among events that are not necessarily related, but in fact, they might be opposite to each other. Consider the following two Microsoft events: the event type 4728 shows that a member was added to a security-enabled global group, whereas the event type 4729 explains that a member was removed from a security-enabled global group. Both of these events have almost similar content and properties, but they are triggered from contradictory activities. The clustering algorithm always inserts them in the same cluster depicting a relationship, due to lack of semantic understanding. Same applies to the event types 4732 (A member was added to a security-enabled local group) and 4733 (A member was removed from a security-enabled local group). Hence, the clustering approach only groups similar items together that do not necessarily represent security identification and mitigation activities. Another drawback is that the clustering algorithms only process numeric data, whereas the event logs contain both numeric and non-numeric entries. The conversion of non-numeric to numeric data would decrease the efficiency of the overall solution.

2.4.2 Automated Planning Techniques

The knowledge in event logs can also be acquired with the help domain model learning. This approach takes a plan of actions (or plan traces) against a particular problem and learns a domain action model to represent the knowledge. The learning process is based on determining the expert actions and the relationship among their objects (i.e. parameters). The domain model is then used by Artificial Intelligence (AI) planners to determine a plan solution for unseen problems. The knowledge in the domain model is required to be sufficiently efficient, so that it can be used for automated reasoning

and construction of a plan for output. Several tools and techniques are available that can construct domain models either manually by human experts or autonomously using machine learning techniques.

For manual construction, several tools are available that provide interface for the design, validation and verification of domain models. For example, an open source tool, called itSIMPLE [46], uses Object Oriented architecture and Unified Modelling Language (UML) to enable the users for building, visualising and modifying domain models. The knowledge is initially constructed in the form of UML, which is then converted to Petri Net graph and shown as domain model. This tools is also capable of analysing dynamic aspects of the modelling requirements, such as deadlocks and invariants. Another tool, called GIPO (Graphical Interface for Planning with Objects) [47], was created to increase the efficiency of domain modelling and validation process. This tool uses its own object-centred language to encode knowledge, domain structure and several other features. It also includes a built-in visual editor that allows the user to inspect and verify the produced plans graphically. Another tool, named EUROPA (The Extensible Universal Remote Operations Planning Architecture) [48], was developed by National Aeronautics and Space Administration (NASA) to tackle real-world problems. It allows an interactive planning process, and provides support in domain modelling and plan visualisation.

Researchers have also produced several autonomous domain learning and modelling techniques. An algorithm, called SLAF (Simultaneous Learning and Filtering) [49], was created to learn the actions using partially observable (unfamiliar and uncertain) domain knowledge. It takes action-observation sequence, states of the world and partial observations of intermediate transition states between action executions as input. As a result, it produces action model that consists of preconditions and effects containing implicit objects and unstated relationships between objects. Another automated tool, called Opmaker [50], takes a partial domain model and a set of training data for planning, and outputs a complete domain action model. An important feature of this tool is the heuristics that are part of the output and can be used to make plan extraction more efficient. The tool automatically determines the intermediate transition states by tracking and generalising the object changes provided in the training set. It was observed during the experiments that some objects have multiple aspects and behaviours, and thus require Finite State Machines (FSM) to completely represent them. A FSM is a

computation model that simulates sequential logic and presents all possible states of an object. With this motivation, a tool called LOCM (Learning object-centric models) [51] was developed that induces the domain model by allowing multiple state machines to represent a single object using FSM characterised by a set of transitions. The extracted domain models only include dynamic facts. A major limitation of LOCM is lack of compatibility with domains having static facts (i.e. remain unchanged throughout the planning process [52]). Recent advancements have started to address this issue, such as in Automated Static Constraint Learner or ASCol [53]. The ASCol exploits a directed graph representation of operator arguments in a plan solution and uses that to discover pairwise static relations among preconditions and effects.

The learning process of the aforementioned tools is dynamic and usually takes additional information, e.g. partial domains and uncertainty factors, for building the complete domain model. The input used is gathered from goal-based solutions and manual observations or experiments. Despite the advantages of state-of-the-art domain learning solutions, a major drawback is the need for pre-existing domain knowledge in the form of plan traces and human assistance. A plan trace is the sequence of actions that represent a solution to achieve a desired goal. Almost all of the existing solutions require plan traces in some form, and as such conducting autonomous learning in a previously unseen application area would first require the construction of a plan trace. In practical and real time domain modelling processes, this is not feasible because creating well-formed plan traces could be viewed as almost as challenging as building a domain model. More specifically, a plan trace can only be constructed once the domain is fully comprehended. In addition, it would also require expert knowledge, something which is not guaranteed to be always available. These issues can be resolved by applying different techniques, like, crowd-sourcing to acquire action models [54] or learning from human demonstrations to perform case-based planning [55]. However, such techniques are expensive and require large amount effort to prevent human-errors and perform conflict resolution. Another significant challenge is that the learning solutions cannot acquire complete and sound domain model if the data sources are noisy and incomplete. This motivates the research challenge of developing an autonomous approach to acquire and use security domain knowledge directly from the available data sources, without having prior knowledge and human expert support. The output plan will be capable of helping experts and non-experts alike for the improvement of security in any new or previously

unseen machines.

2.4.2.1 Domain Representation Languages

After acquiring the domain knowledge, whether through manual or autonomous approach, a variety of modelling languages are available that can be used for encoding domain knowledge. The following sections explain some of the existing domain modelling languages.

Stanford Research Institute Problem Solver (STRIPS) – The STRIPS language was developed in 1971 [56] and is the base for most of the latest languages in use today. A STRIPS planning problem contains the following three components: (1) initial state, (2) goal state and (3) declarative actions containing preconditions and post-conditions. This language only supports positive literals along with their conjunctions (combining multiple literals into a single literal). The capability of defining preconditions in each action is the major accomplishment of STRIPS as no assumptions are made regarding the structure and validity of domain, and also allows the domain-independent planning. In early days of AP, STRIPS was considered as the most popular representation language.

Action Description Language (ADL) – The ADL is considered as a successor of STRIPS and was particularly developed for robots in 1987 [57]. An ADL contains the following four components: (1) actions, (2) optional parameters for each action, (3) optional clauses for preconditions and (4) add, delete and update parameter list that is not part of the action schema. The ADL supports both positive and negative literals and their conjunctions, unlike STRIPS.

Planning Domain Definition Language (PDDL) – Considering the existing literature on domain model generation, the Planning Domain Definition Language (PDDL) along with its variants are the best-known languages for modelling domains and planning problems. The PDDL is a successor of STRIPS and ADL and was developed for the planning competition with the aim of having a single and common domain representation language. Many planners support PDDL due to its wide usage and acceptance in the AP community. The PDDL is a standard, action-centred language and is also based on first-order predicates. Some of the important versions of PDDL are explained in the following:

1. PDDL 1.2 – This version of PDDL was the official language of first and second International Planning Competition (IPC) in 1998 and 2000, respectively. The PDDL 1.2 [58] divides a planning problem into the following two components: (1) domain action model to represent knowledge and (2) a problem instance to describe the problem. It is suitable for classical planning as it includes few additional features, such as object types and conditional effects;

2. PDDL 2.1 – This version was the official language of third IPC in 2002. Appendices B.1 and B.2 provide detailed schemata of domain model and problem instance, respectively. The PDDL 2.1 [59] is more equipped to represent real-world problems as it introduced numeric fluents, plan-metrics and durative-actions. A numeric fluent (also referred as a function) is a type of state variable or predicate that is used to model non-binary resources in the form of real numbers, such as distance, volume, weight etc. These fluents can be used in both precondition and effect definitions. In precondition, a fluent simulates an if-statement, where the comparison between fluents and real numbers is performed with the help of following symbols: less than or equal to ($\leq$), less than ($<$), equals to ($=$), greater than ($>$) and greater than or equals to ($\geq$). In effect definition, a fluent can be used as a then-statement that performs the following arithmetic operations: addition ($+$), subtraction ($-$), division ($/$) and multiplication ($*$). Other operators are also available that can modify a fluent, such as *increase* or *decrease* value by certain amount and *assign* a new value. The plan metrics enabled the quantitative evaluation of plan solutions using numeric fluents. This increases the capability of goal-driven planning algorithms and performs plan optimisation through minimisation and maximisation criterion. The durative-actions allow the simulation of temporal aspects of plan solutions. The time is encoded with respect to action's duration, where preconditions and effects are defined for either beginning or end of the action. The preconditions can also be defined for the entire duration of the action;

3. PDDL 3.0 – This version was the official language of fifth IPC in 2006. The PDDL 3.0 [60] improved the previous PDDL versions by adding numeric constraints to perform arithmetic operations. It was the first version, where temporal planning was introduced by adding features to durative actions. It also includes hard and soft constraints, where hard constraints must be true during the plan execution

while soft constraints (or preferences) if found to be true, will increase the quality of plan;

4. PDDL 3.1 – This version was the official language of deterministic track of sixth and seventh IPC in 2008 and 2011, respectively. The PDDL 3.1 [61] introduced object fluents, where the range is not limited to numbers but can also involve any object type. The syntax of PDDL 3.1 is significantly expressive in terms of semantic;

5. Other versions – Probabilistic PDDL (PPDDL) [62] and PDDL+ [63] are also among useful extensions of PDDL. These two languages have been applied to several real-world problems. The PPDDL was presented in the probabilistic track of fourth IPC and incorporates probability distribution in actions. PDDL+ is specially designed for continuous planning problem with respect to discrete-time, in order to encode continuous processes. Another domain representational language is New Domain Definition Language (NDDL) [64], which was developed by NASA in 2003. The NDDL mainly focuses on space applications. It uses object fluents and is capable of modelling continuous planning tasks. A major disadvantage of NDDL is the lack of supporting planners.

2.4.2.2 Searching Techniques

After representing the knowledge in a domain action model, Automated Planning (AP) algorithms are applied to select and organise purposeful actions in achieving expected outcomes [1]. Several AP algorithms are implemented in software solutions, known as automated planners. Every planner employs a searching technique [65] to find a sequence of actions from initial to goal state. The searching can be classified into either uninformed or informed approach that explores or traverses the state space to determine one or more solutions P by using a forward chaining mechanism. The state space is a set of all available states that are organised in a graph structure, where the nodes represent states, the edges are indicative of domain actions (representing state transition) and the path from initial to goal nodes represent a plan.

The uninformed search – This approach traverses the graph without using any additional knowledge, and is based on two basic algorithms: breadth-first and depth-first

search [66]. The breadth-first search (BFS) first traverses all the nodes on the same level before moving to the next level. The BFS repeats this process until there are no more states left to traverse or the goal state is reached. The space and time complexity of BFS is $O(b^d)$, where d is the depth of goal node and b is the number of successors of all non-goal nodes (branching factor). The BFS is complete and optimal but uses a large amount of space. On the other hand, the depth-first search (DFS) first selects a node and completely traverses it until either the solution is found or there is no successor of the current nodes. The DFS will then backtrack one level in the path and traverse the successor node in the same manner. The time complexity of DFS is $O(b^m)$ and the space complexity is $O(b * m)$, where b is the branching factor and m is the maximum depth of any node. The DFS is neither complete nor optimal but uses less space and resources.

The informed search – This approach makes guided decisions based on some prior knowledge or a heuristic function other than the domain model. The heuristic function is used to obtain the optimal path between two nodes of the graph. It can also enable a 'greedy' search, where the nodes that appear closest to the goal are traversed first in order to find a quick solution. Several algorithms are available to perform the informed search, such as A* [67] and branch-and-bound optimisation [68]. The A* algorithm is one of the most common algorithms. The aim of this algorithm is to perform the best-first search and avoid traversing nodes that appear to be expensive. On current node n, this is done by an evaluation function $f(n) = g(n) + h(n)$, which determines the cost of the previous path $g(n)$ and an estimated cost of the remaining path all the way to the goal $h(n)$. The other informed search algorithm, branch-and-bound optimisation, maintains lower and upper bounds on the global objective value over a given region. The upper bound B denotes the best solution found so far during the search. The lower bound is calculated on traversing a new node by a heuristic function I that estimates the cost of the path currently being explored. If $I > B$, then the algorithm removes the new node from path as it will not result in a better solution with a lower cost than B. Most of the new planning algorithms employ heuristic functions to improve their accuracy, efficiency and productivity [69], hence employing informed search approach.

2.4.2.3 Applications Of Planning In Cybersecurity

The plans are generated by considering optimisation constraints defined in terms of quality and quantity. It is demonstrated through experimental analysis that such (computer-generated) plans provide better and more efficient security solutions depending on the quality of domain model knowledge [70] because the planner search and explore a wide array of possibilities before finding the most feasible solution. Therefore, AP has become an integral part of numerous security-related areas, which is also evident from the literature discussed in the section below.

There have been successful exploration of the use of AP in producing attack plans for penetration testing. One of the earlier work involves utilising classical planning to generate hypothetical attack scenarios that can exploit a web-based system [71]. This study simulates realistic adversary courses of action, with the main focus on malicious insider threat. The presented domain model includes 25 objects (hosts, processes, programs, files, etc.), 124 propositions representing information about system (configurations, vulnerabilities, etc.) and 56 actions to denote the adversary's objectives. The problem instance is based on $200 - 300$ facts of the underlying system. A classical planner, named FF-Metric [72], was used to extract the plan of attack actions. The domain model was constructed by human expert using a semi-manual (i.e. employing assistive tools) approach. The study also presented a post plan processing tool, which was written in Perl script, to convert the planner output in a human-readable format.

Another research study [73] used planning to assess the degree of network security. It starts by transforming the attacks into domain models, which represents the requirements and exploits, whereas the information about system, such networks, ports, operating systems, machines and services are encoded in problem files. The proposed approach analyses the entire network using the domain mode, which has up-to 1800 actions and hosts 700 exploits. A similar research work [74] presented a solution that can generate sequences of actions to compromise a network in the low cost, shortest path possible. The actions comply with the network environment and configuration and are aimed at exploiting a series of vulnerabilities for adversarial gain. A commercial tool (*Core Impact*) has also been developed that uses AP to generate possible attack plans and is capable of conducting real-time penetration testing [75]. The tool extends the domain model to encode probabilities of attacks. The tool also builds $AND - OR$ trees

to determine all possible attack paths against a certain asset. The tool is claimed to be efficient with respect to execution time and amount of network traffic. The computational complexity is described as *O(n log n)*, where n is the total number of actions in domain file.

One of the main issue in these solutions is the lack of ability to handle incomplete knowledge and scalability limitations. Partially Observable Markov Decision Processes (POMDP), can be used to eliminate the problem of incomplete knowledge and uncertainties [76]. POMDPs are capable of ranking the domain actions based on expected reward, which is composed of asset value and time/risk of detection to accomplish the desired goal. Further research [77] emphasises how to produce better attack plans for a particular machine within the shortest time possible. The proposed solution uses intelligent vulnerability scanning actions with the help POMDPs and discovers feasible attacks for each individual machine. Despite all the advantages of POMDPs, constructing them is a complex task and they also require large computational resources. It is also difficult to calculate the probability values for every attack scenario. To address the scalability issues, a study [78] proposed a middle ground between classical planning and POMDPs, called MDPs. The probability value depends on the level of attacker's uncertainty in launching that particular action. The values will remain constant throughout the planning process and assigned regardless of environment or configuration.

2.4.3 Association Rule Mining Techniques

Considering the importance of creating a domain model in an autonomous manner, this research pursues potential Association Rule Mining (ARM) techniques that are unsupervised and can discover relationships among a diverse set of event log entries. ARM is a sub-domain of data mining that refers the method of identifying patterns to reveal strong/frequent co-occurrences in the form of correlation rules, containing antecedent and consequent, among seemingly unrelated items [79]. The first ARM algorithm [80] was created to identify regular co-occurrences between the products in large transaction data. The data was recorded by point-of-sale systems used in supermarkets. This initial approach could only have one consequence statement in the rule, i.e. it can determine $X \cap Y \rightarrow Z$, but not $X \rightarrow Y \cap Z$ types of rules. Following this, two new algorithms were introduced called Apriori and AprioriTid [81], which were based on breath-first search.

A detailed working example of Apriori algorithm is provided in [82]. The main target of these algorithms was to improve the performance of ARM process by reducing the execution time. Apriori was the first algorithm to pioneer efficient pruning scheme and prevent the exponential growth of possible frequent itemsets. The algorithm iteratively generates sets of items in increasing arity; singletons, pairs, triplets etc. The general idea to increase efficiency was based on the Monotonicity Principle [83]. Given an itemset $ABCD$, first determine all of its subsets ABC, AB etc. If a subset, say ABC, does not generate a rule, remove it as there is no need to process its further subsets. Similarly, if any itemset is found to be frequent, then every subset of the itemset will also be considered frequent. This property is known as the Apriori Principle. Another approach was introduced to further increase the performance of ARM by storing the itemsets in a hash tree format [84]. The proposed algorithm, called DHP (direct hashing and pruning), uses a hashing technique to trims unnecessary itemsets in every iteration, thus reducing the number of transactions for further processing. According to a study [85], there are 38 measures that can be used either individually or combined to improve the quality of rule mining process. Some of the measures are support, confidence, lift, recall, precision, specificity, etc. The proposed solution in this research uses the ARM technique, which relies on support, confidence and some 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.

ARM has been widely exploited in many different application areas. A research study proposed a tool, called LogMaster [86], which uses ARM to find the correlation between failure and non-failure events in the logs data, and predict future crashes of the system. This tool starts by pre-processing the events data and applies the Apriori algorithm to mine correlation rules. The extracted rules are collectively represented as an event correlation graph. After that, the tool uses events timings and statistical measures on the graph to build event prediction model The LogMaster was tested on three sets of Cloud event logs, with the average precision rate of 83.66%, 81.19% and 79.82%, respectively. A similar study, which is only based on frequent itemset mining, has been used for knowledge discovery in the event logs [87]. This study presented a solution, called AllMining, which consists of the following three steps: pre-processing of event logs data, discovering frequent co-occurrence in events and pruning the irrelevant frequent itemsets increase the quality of knowledge discovery and decision-making process. The

evaluation of AllMining shows that it is scalable and efficient tool, and provides better coverage of knowledge. Researchers have also proposed a variant of ARM that can identify irregular rights/permissions in a file system [88]. This solution finds the rules with least frequent or irregular co-occurrences, as oppose to finding strongly correlated rules like in the most applications of ARM. This allows them to identify anomalies or rather suspicious permissions that should be removed to improve the protection of user data.

A research study has proposed the use of ARM in analysing server logs [89]. It provides a new variant of FP-Growth, called a Long Sequence Frequent Pattern (LSFP) algorithm. The LSFP takes the log files, produces association rules as a directed graph and creates a description or timeline of the user behaviour. According to the authors, support value has a direct impact on the quality and efficiency of the overall solution. Another research work used an ARM technique to detect phishing URLs [90]. First, a set of features is produced that can help in distinguishing between the legitimate and phishing URLs, such as length of the host URL, number of slashes, dots in hostname, etc. After that, Apriori algorithm is applied to find interesting patterns and combination of features in the form of rules. The developed solution was tested on a dataset containing 1400 URLs and achieved 93% accuracy. A recent paper has used ARM to categorise Darknet network traffic streams to gain a better understanding of global trends in cyber-attacks [91]. This paper considers two cases; correlation between attacking hosts and destination ports, and the correlation between Darknet traffic probing sensors. The ARM discovered some significant rules as they comply with the behaviour of known botnets. Therefore, this paper claims that the mined rules can facilitate in building secure strategies.

It should be noted here that the existence of association relationship among mutually exclusive items of the dataset does not necessarily imply the actual connection [92]. The rules are subjective and usually applicable to the given data (i.e. the results are local, not universal). In other words, a rule is a just depiction of strong correlation between the antecedent and consequent data items. Hence, there is a need to improve the quality and confidence of the rules. This is can be achieved by adding an extra measure to determine meaningful correlation, as evident from many studies. One such measure is temporality or time-based ordering, which can be used to identify and extract interesting association rules among items [93]. It provides complementary evidence for the relationship to exist. For example, consider event X (antecedent) and event B (consequence) are in

associative relationship. Finding whether event A always happens before event B can help in establishing a higher confidence. Several research studies are available that developed different methods for temporal ARM and obtained quality results. One of the earlier work proposed using temporal metric as an interestingness measure for ARM (based on a survey) and demonstrated the numerous benefits in terms of relevancy and quality of knowledge discovery in rules. It also showed that the incorporation of temporal dependencies in ARM can group and (re)arrange rule items into correct semantic sequences [94]. Later on, a research work developed a new algorithm, called T-Apriori [95], which integrated a time-based constraint in Apriori algorithm and used it for analysing ordered ecological events set. The algorithm requires an additional layer of data pre-processing using K-Means clustering algorithm, which reduces the overall performance. Also, according to the authors, the efficiency of the algorithm has not been tested on larger datasets. Another study presents an algorithm, called ARMADA [96], which uses time dependency to discover sequential frequent patterns. The algorithm was tested on large, interval-based data that was constructed synthetically. The authors claim that the quality temporal association rules were extracted and provided richer knowledge with respect to semantics.

A recent study has proposed a new mining algorithm for time-series dataset, called temporal association rules with frequent itemsets tree (T-FS-tree) algorithm [97]. This algorithm mines frequent itemsets and builds a FS-tree structure in parallel, hence effectively reducing the computational cost by skipping the candidate itemset generation phase. The algorithm outputs interesting temporal association rules, which are acquired by traversing and pruning the irrelevant rules in the tree. Another work proposed a Multi-level Intensive Subset Learning (MISL) algorithm [98], which modifies frequent itemset mining to include a temporal metric. The algorithm uses temporal metric to identify and remove such rules that have occurred by chance. It was successfully used on financial time series data and identified profitable trades by generating association rules, which lead to uncover hidden knowledge. Another study takes presents a different technique that first generates a complete set of association rules from the given time-stamped data, and then convert them into temporal-association rules based on the co-appearance of sequences [99]. This approaches requires input of states and events of interest. It has been successfully tested on live systems where data belonged to fish movement in a river. Another correlation technique for heterogeneous data has also been

developed that connects the security events of a system based on clustering, properties and temporal precedence [100]. The output is presented as a graph, which shows both direct and indirect connection of events. For testing, the data was collected from multiple sources and the resultant graph contained 1309 nodes identifying many malicious activities regarding malware, network attacks and user behaviours. Another research study presented a solution for assisted living using a temporal metric in Apriori algorithm that can be used for activity reminder and abnormal behaviour detection systems [101]. This solution recognised temporal relations of user activities, such as start time and duration, and used that to construct a predictive activity model.

Despite having significant advantages, the existing ARM approaches are unable to yield a suitable solution for identifying temporal correlation rules in raw data streams using a fully automated mechanism. They usually either rely on the manual input by human or operate in a determined, constrained, specific and predictable environment to extract the rules. Some of the solutions require additional configuration and processing layers, which might not suit large-scale and complex models. Furthermore, the need for manual input again leads towards the human assistance and expertise, which as discussed before lacks efficiency and are in short supply. Another significant lacking in the existing techniques is the incapability of finding the initial (starting) item of a temporal sequence/pattern of multiple correlated events. As the results are shown as a graph or cluster, they cannot provide the basis to develop a domain action model.

2.4.4 Causal Inference Techniques

The existence of correlation does signify a strong statistical relationship among the linked items (or events in our case), but it is not always true [102]. The relationship between events can be made certain by finding evidence of causation. A cause and effect connection demonstrates that the events are dependent on each other given certain conditions and are defined with the help of Markov chains [103]. A Markov chain is a sequence and description of random events that cause the system to transition from one state to any other state. It also includes the probability of each transition, which depends on the state reached due to the previous event only. A number of algorithms have been devised that can find causality among items. The process of determining causal relationships can be broadly classified into two main categories: *CCC* and *CCU* triplets causality [104].

Both techniques are based on induction and statistical probability principles. Assume there are items: A, B and C. The CCC mechanism considers all three possible pairs of items as correlated and one variable is already known 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 remaining B causes C. On the other hand, the CCU mechanism is that only two pairs are in associative relationship, and one pair is not correlated. For example, if the pairs (A, B) and (A, C) are correlated and (B, C) pair is not in association, then B and C cause A. The CCU causality mechanism is deemed as better in performance, while the CCC is better in producing quality results.

Many techniques have been introduced in previous years that can discover causal relationships. Most of the algorithms are based on either Bayesian [105] or Constraint-based [106] approaches. Both approaches are based on Markov chain but differ in theory and practice. The Constraint-based approach can make categorical decisions about the conditional dependence and independence constraints among the data items, whereas the Bayesian approach weighs the degree in terms of probability to which these constraints hold. If two items are conditionally dependent (or not conditionally independent), it means they are in cause and effect relationship. Any Constraint-based approach is based on two steps: conduct statistical tests to determine conditional (in)dependence among the data items, and use the tests to define the types of causal connection that exist among the items. This approach uses different kinds of statistical tests to reduce the complexity and complete the process in a reasonable time. Furthermore, the Constraint-based solution have been widely utilised in several real-world problems as they are more generalised. On the other hand, Bayesian solutions find all possible causal structures and use a probabilistic framework to determine if 'Event A cause Event B'? This approach requires user-specified probabilities in the graph structure. If it is not feasible for the user to provide this information, then non-informative priors can be used. For a large and complex observational data, Bayesian solutions will require high computation power to identify and process all causal structures. Moreover, the incomplete or hidden variables may also make it difficult to directly observe the data and produce accurate results.

One of the earlier research studies presented an algorithm, called Local causal discovery (LCD) [107], which uses a Constraint-based approach to determine pairwise causal

relationships in a completely observational data. The LCD algorithm uses CCC mechanism (but not the CCU) to conduct independence tests between all data items. It uses a function, called $Independent(A, B, C)$, which implicitly scans the entire set of relationships R to determine whether A is independent of B, given C. If found independent, then the function returns true, or else, it returns false. This function is applied to each pair of items and causal structures are represented as a graph. A recent study presented an algorithm, called Logical Causal Inference (LoCI), that conducts minimal conditional (in)dependent tests and then convert them into logical statements to form initial causal relations. After that, the algorithm combines all statements using aggregation, elimination and basic properties of causality to output a complete causal structure. Another study presented a Constraint-based algorithm, called Markov blanket/collider set (MBCS*) [108], which starts by finding a Markov Blanket (MB) of each item to determine all items (parents and children) that shield this particular item from the rest of the network. So, instead of conducting conditional-independence tests for all pairs of items in the data to infer causal relationships, it only considers MB of each item and provides better results in terms of accuracy and efficiency. Several Constraint-based approaches are also developed for learning a single, collective causal structure from joint observational datasets. One simple algorithm for this purpose is Structure learning using prior results (SLPR) [109]. The algorithm starts by finding the causal structure for each dataset individually and then combines them by removing common edges to get a full graph. After that, it reapplies causal inference mechanism to the graph and output a final causal structure.

A research work presented a method for discovering causation through Bayesian approach by using a mixture of observed (deterministic) and experimental (non-deterministic) data [110]. The proposed model was tested on a manually created dataset of potential anaesthesia problems in the operating room that contains 37 nodes and 46 edges. An improvement made by this solution is the automatic assignment of initial probabilities, which are then used to build a Bayesian network. After that, it used joint probability distribution to successfully infer the causal structures and parameters among randomly selected item-pairs. A variant of LCD algorithm has also been proposed, called Bayesian Local Causal Discovery algorithm, which reduced the amount of false positives and increased efficiency [111]. It uses a Bayesian scoring metric instead of conditional independence testing. The results are modelled as a probabilistic graph of connected items,

known as Bayesian network. The Bayesian network satisfies Markov chain property and the availability of probability distribution allows the efficient inference between two random items. After that, it applies a heuristic-based greedy search in the Bayesian network to derive Markov Blanket (MB) for each node. By limiting the causal learning to the MB of each node, the algorithm significantly reduces the size of probability-based inference calculations, and thus have improved accuracy and efficiency. A similar research study presents two hybrid (Bayesian- and Constraint-based) algorithms CD-B and CD-H that can discover causal relationships from observational data [112]. The CD-B algorithm employs a greedy search to determine the MB of a node, and then use it in a scoring method to identify the parents and children. This process is repeated for every node and a global Bayesian network is constructed to discover all causal structures. On the other hand, the CD-H algorithm is based on a two-step mechanism to perform conditional dependence and independence tests and determine the parent-child relationships of all nodes. The rest of the process is similar to the CD-B algorithm.

There are also other causal mining techniques that are not based on Bayesian/Constraint-based approaches. An algorithm, called causal association rule discovery [113], presented a correlation mining approach to determine causal rules in observational data. It starts by extracting all association rules, and for each rule $(A - B)$, it hypothesises that the left (A) causes right (B) item. After that, it applies retrospective cohort studies mechanism to find odds ratio for each corresponding hypothesis and discover persistent causal rules. Another study presents a Relational causal discovery algorithm [114], which uses automated relational blocking to determine causal relationships, instead of statistical tests. The relational blocking is essentially a manual technique that divides the data into unrelated groups based on certain criteria (blocking factor). In the proposed algorithm, groups are defined based on a graph structure that aims at reducing variation and adjusting for common causes. The groups are then used to determine causal structures. However, unlike conditional (in)dependence tests, the relational blocking is not capable of inducing dependence when discovering common effects. Apart from the algorithms, certain mechanisms are developed to obtain evidence of a causal connection between a presumed cause and an observed effect, for example, the Bradford Hill [115] and Granger [116] models. They claim that a causal rule should satisfy the following factors: association, specificity, consistency, plausibility, temporal order, coherence and experimental evidence. Another approach to infer causality is presented by Popper,

which is based on three considerations: temporal precedence, dependency, and no hidden variables [117]. This approach constructs incremental causal network and has been successfully exploited in the unbounded data streams of events to build causal networks.

2.5 Literature Analysis

This section provides a brief discussion on the current state of knowledge acquisition systems:

1. Several data sources are available that can provide security knowledge to the non-expert users;
2. The manual or assistive techniques are heavily reliant on the manual processing of knowledge, which requires human experts, domain-specific experience, error and conflict management and continuous maintenance in terms of updating and constructing new knowledge. These techniques might also be counterproductive for large-scale knowledge acquisition as it would be time, cost and resource-intensive, and demand additional expertise in building, representing and storing complex patterns for utilisation. Such methods are not feasible for our research as we are aiming to acquire knowledge in an automated manner that does not require any human intervention;
3. Most of the security-based applications of manual or assistive event correlation techniques are in monitoring and fault/anomaly detection;
4. The applications of automated planning mostly lies in the domain of offensive security, i.e. penetration testing and attack planning;
5. The existing automated technologies have shown strong theoretical foundation, but are not fully equipped to process the data sources in generating actionable intelligence of higher quality and accuracy. The solutions are either too simple or difficult to reach out to the necessary audiences. Also, they require a high use of expert knowledge and results in high costs and poor scalability; and
6. The existing literature shows that no research has been conducted to extract a set of security-related actions performed on a system and utilise them to create plans for improving the security of vulnerable machines, all without any human support.

2.6 Chapter Summary

This chapter presents the relevant background theory that establishes the basis of this research. It starts by introducing main areas with the emphasis on automated knowledge acquisition, representation and utilisation. This chapter aims to provide a reader with the background of contributions made in this research, along with surveying past research in the closely related areas. This chapter starts by investigating the data sources that can provide useful security knowledge. After that, it explains some manual and assistive knowledge acquisition techniques along with their lacking and drawbacks. Further ahead, the chapter explores the state-of-art research on the role of Automated Planning in cyber security-related applications, correlation and causal rule mining techniques, and how they are extensively used for automated knowledge acquisition. All references are provided against discussed literature. The chapter does not provide detailed review and analysis of all aforementioned techniques but provides a context of important concepts to aid the understanding of the contribution of this thesis.

Chapter 3

Automated Knowledge Acquisition

Automated knowledge acquisition (KA) is an alternative to the laborious procedures of collecting the right knowledge from given data sources. This chapter explains the novel mechanism of acquiring 'rules' from Microsoft security event logs that contains the expert knowledge. The entire process is performed in an automated manner, i.e. computer-mediated discovery and representation of knowledge that allows the elevation in the performance, effectiveness, efficiency, coverage and continuous updating of a knowledge-based system. Following are the five steps and their corresponding components that have been developed as part of this process, and also shown in Figure 3.1:

1. Data pre-processing:
   (a) Identify and remove routine events from the given event log dataset; and
   (b) Convert the remaining event log entries to an object-based model.
2. Association rule mining:
   (a) Determine both minimum and maximum support values from the object-based model;
   (b) Identify object-based rules from the model using the support values in an association rule mining algorithm; and
   (c) Determine the event-based rules, such that any two events are associated by the object-based rules.

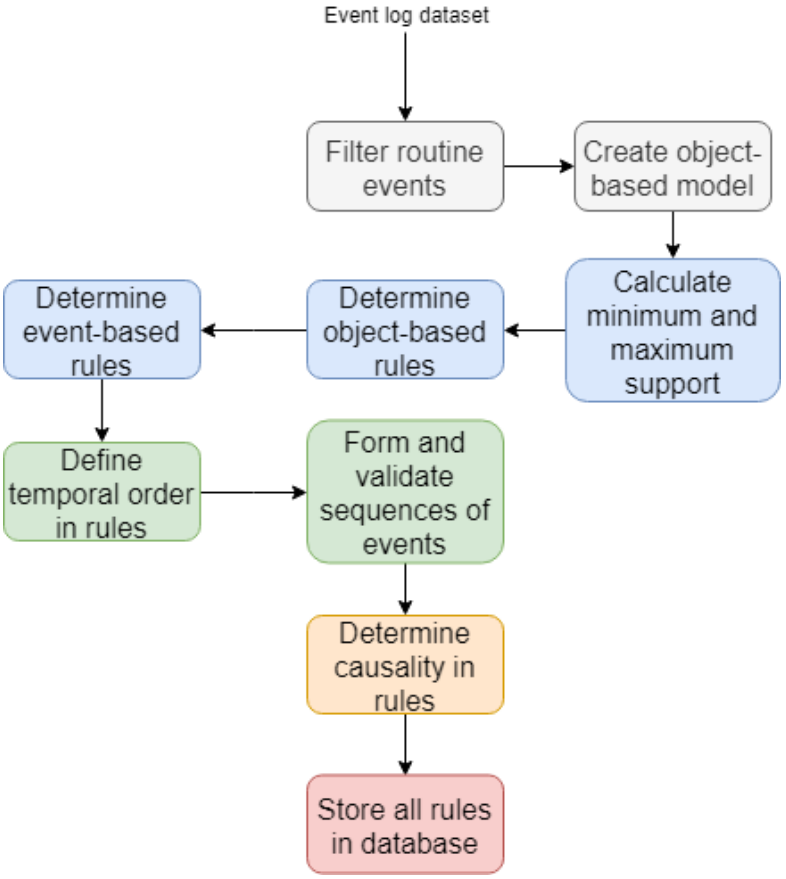

FIGURE 3.1: Summarised steps of proposed automated knowledge acquisition process, where same coloured components belong to a single step.

3. Sequences of event relationships:

   (a) Improve the quality of relationships based on the temporal order of events, alongside discarding insignificant relationships; and

   (b) Combine and expand the individual relationships to form and validate sequences of events.

4. Determining causality:

   (a) Convert the sequences of events into directed acyclic graph and infer a causal rank for each relationship.

5. Storing Rules:

   (a) Store all relationships in a database along with all relevant information.

3.1 Data Pre-Processing

This section presents the pre-processing phase of security event log entries. It has two steps and prepares the data for passing to 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.

3.1.1 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 to denote that the maximum log size is reached and the operating system starts saving the new event log entries in another file, as an alternative to over-writing the existing one. The description of event 1105 is 'Event log automatic backup' and it only contains a single object-value pair, such as *File: C:\Windows\System32\Winevt\Logs\Archive-Security-2019-02-11-18-37-03-002.evtx*, to display the file name where new entries will be saved. Similarly, the event type 4608 does not show any security related activity. The description of event 4608 is 'Windows is starting up' and has no object-value pair as it is only logged for information.

Although the 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 research, such routine events are considered as *noise* due to their relatively high frequency of occurrence, lack of information and their strong relationship amongst a large volume of system users and resources. Also, 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. Furthermore, when the rules are encoded into a domain action model for knowledge utilisation as described in the next Chapter 4, the resultant action plan for the identification and ramification of security issues loses conciseness, hence creating unnecessary difficulty for the non-expert users.

To remove routine events, the first step 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 a live system:

$$FD = \{152, 35, 158, 2148, 10, 36, 4, 4, 2121, 32, 8, 44, 44, 23, 1, 1, 1, 8, 2, 1, 2, 1, 26, 1, 2, 2, 5, 152, 1, 1, 1\} \tag{3.1}$$

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 detection and elimination of the routine events in an automated manner, Mean (μ) and Standard Deviation (s) values can be used. A μ is the central value of any given set, more specifically, it is the sum of elements divided by the number of elements. The s is a measure of spread of elements in a set [118], and determines that how much they differ from the mean value (μ). A low s depicts that the elements are closer to the μ, while a high s value shows that the elements are distant from the μ. In other words, lower s means that the elements do not have substantial numerical difference among each other, while higher s means that the elements are dispersed over a wider range of values. The formula of s is shown in Equation 3.2:

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2} \tag{3.2}$$

Where N denotes the number of elements, x_i is the i_{th} element and $\overline{x}$ is the average of FD. It is possible that the given set of event log entries does not contain a complete record of security-related activities due to insufficient computing resources for event logging mechanism, memory or storage corruption while saving continuous stream of events, lack of defining time period in filtering events etc. Hence the proposed solution uses Sample Standard Deviation to form an estimate of larger population and output a generalised result. Instead of using N, which is used in the Population Standard

Deviation, it uses $N-1$ in the denominator of Equation 3.2, which is known as Bessel's correction [119].

The ratio of the s to the μ is called the coefficient of variation (C_v), which is a measure of variability [120]. If the value of C_v is greater 1 (or 100%), it means the set contains one or more outliers [121], i.e. the frequency of certain event types is significantly higher or lower 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 3.1, 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 3.3.

$$FD_1 = \{152, 35, 158, 10, 36, 4, 4, 32, 8, 44, 44, 23, 1, 1, 1, 8, 2, 1, 2, 1, 26, 1, 2, 2, 5, 152, 1, 1, 1\} \tag{3.3}$$

3.1.2 Preparing Object-Based Model

To enable a structured mechanism of processing event log entries for automated knowledge acquisition, it is necessary to extract meaningful parts of each event and model all entries into a uniform format. In this research, the normalisation of event log entries is called as 'object-based model'. Each event in the object-based model is represented in terms of the system objects (or properties) found in the entry. These could, for example, be account names, machines names, system resources, permission levels, security identification etc.

In the following discussion, D is used to model a 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:

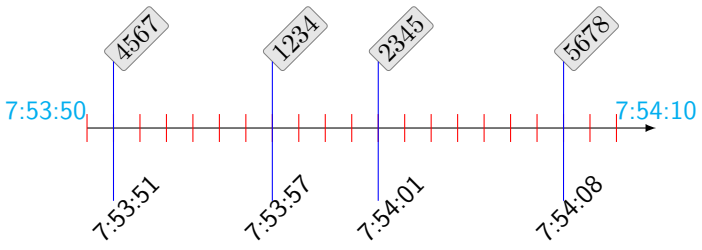

FIGURE 3.2: A visual time line representation of D covering a time span of 20 seconds.

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

In this trivial dataset, each entry consists of an event type (e.g. 4567 and 1234) and 4 objects (e.g. User1 and Win7). A sequential timeline of the dataset is presented in Figure 3.2, where the events occur over a 20 second period. Both event type and objects involved are necessary to understand the occurrence of any event.

It should be noted that the real time event log entries are much more diverse than the presented example. They contain large number of objects and require computing-resource efficient solution for processing. One example of a live entry from Microsoft system is shown in Figure 3.3. This entry contains 33 object-value pairs and is typically generated while modifying the user-group security policies. Each entry is tagged with a time stamp as well to denote creation time. The events only occur over the duration that a system is running and they appear in sequence; although, simultaneous execution of multiple processes can generate intricate entries.

As mentioned before, it is fairly straightforward to acquire the object-value pairs from a structured event log dataset. However, there are some event logging mechanisms that store each entry in a textual format as it is easier to create and provides better readability for the users. In that case, text mining approaches will be employed to analyse the data from different aspects and then extract properties (or features) for constructing an object-based model. Several text mining techniques have been gathered and reviewed in this book [122]. These techniques use a standard representation model for text, called

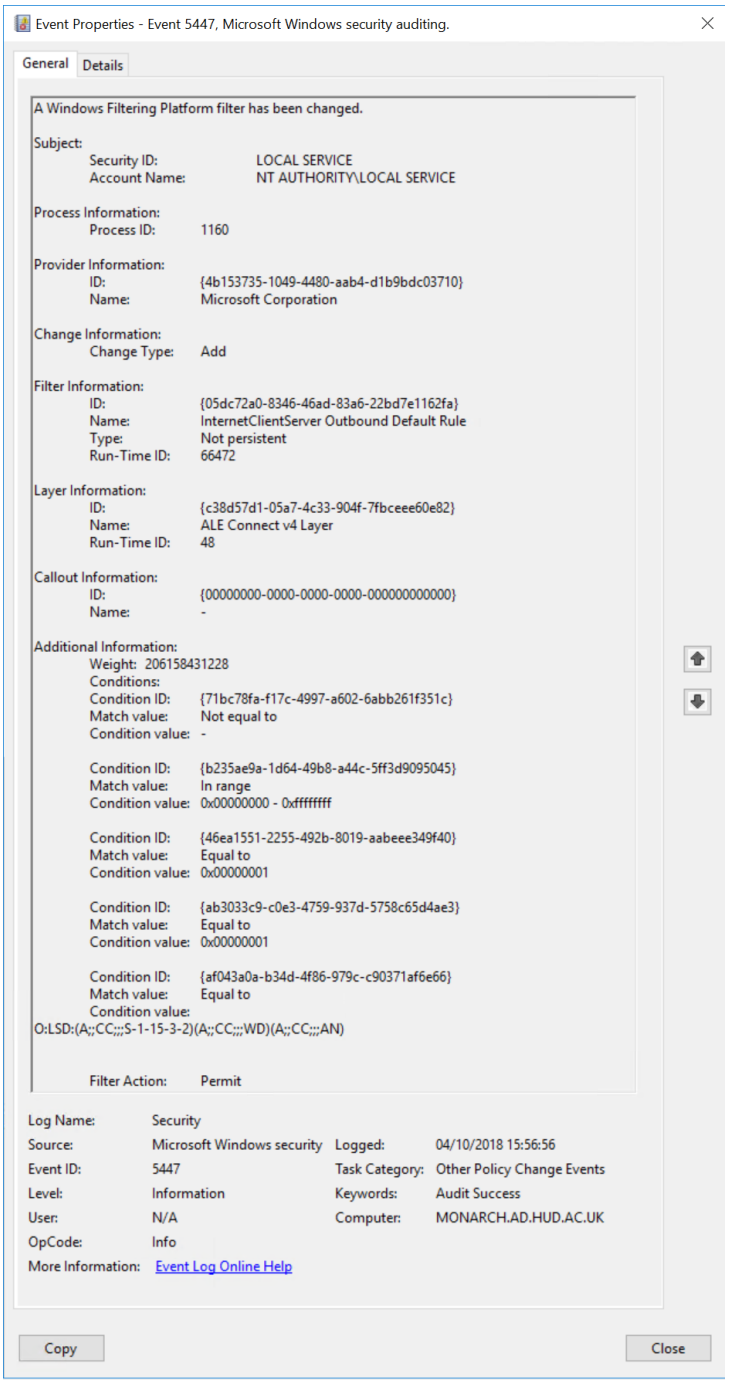

FIGURE 3.3: A live entry of event type, 5447, acquired using the Event Viewer application of a Microsoft system. It shows the description of event, list of all object-value pairs and other relevant information.

bag-of-words. This model disregards grammar and word order, and produces a two-dimensional binary vector based on word appearance. The vector can be processed by several types of statistical, machine learning and natural language processing techniques for extracting the features.

3.2 Association Rule Mining

As discussed in the literature review (Section 2.3.3), there is a potential of using association rule mining (ARM) to identify relationships amongst the objects of event log entries. It is evident from the evaluation of existing solutions that the ARM can successfully discover quality knowledge from all the data streams. ARM is an unsupervised process [123], and employed as a method for explaining, analysing and illustrating 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.

3.2.1 Object-Based Association Rules

The purpose of object-based rule mining is to identify such objects that are likely to occur together. The premise here is that if the objects are co-occurring, the respective events will co-occur too. The proposed solution uses Apriori algorithm [81] for ARM. The association rules are discovered within a tabular dataset of objects that uses different measures to mine interesting results. ARM takes two fundamental steps to extract usable rules:

- *Frequent itemset*: scan the entire dataset to discover the set of all items that appears more than a certain number of times. The number is pre-defined by the user; and
- *Correlation*: identify co-occurrences among the sets and their subsets of frequent items. The degree of which a certain group of items have occurred together defines the strength of their relationship.

The reason behind using Apriori algorithm is that it performs exhaustive search in the dataset to produce large number of frequent itemsets [124], which consequently results in finding complete set of correlation rules. Consider the set D of event log entries and the set I of total unique objects from Section 3.1.2. The fundamental idea behind generating correlation rules is to determine frequent co-occurrence of objects. They identify different objects in event log entries that have appeared multiple times together. The following three steps are taken in order to discover interesting relationships [125]. 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 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 objects, i.e. $X \cap Y = \emptyset$. $X \rightarrow Y$ means whenever E contains X then E probably contains Y too. In the continuing example, an AR would be $\{User1\} \rightarrow \{Win7\}$.

Step 2 (Support of an AR). The support ($supp(X)$) of an AR is defined as the percentage of event log entries that contain both X and Y. It can also be construed as the probability $P\big(X \cup Y\big)$. The support value is also known as statistical significance and ranges between 0.01-1. The example AR has a support of $1/4 = 0.25$, since it occurred in 1 out of 4 event log entries.

Step 3 (Confidence of an AR). The confidence of an AR is defined as the ratio between the number of transactions that contain $X \cup Y$ and the number of transactions that contain X. It can also be defined as $P\big(X \cup Y | X\big) = P\big(X \cup Y\big)/P\big(X\big) = conf(X \rightarrow Y)$. The confidence value ranges between 0.01-1. In the example, the confidence ($supp(X \subset Y)/supp(X)$) of rule $\{User1\} \rightarrow \{Win7\}$ would be equal to $0.25/0.5 = 0.5$. This shows that for 50% or 2 of the 4 entries that contains $Win7$ are related to $User1$.

Researchers have demonstrated that the ARM technique generates a high quantity of rules and has a complexity of $O(N^2)$, where N is the total number of unique items in the given data [126]. In our case, N is the total number of unique objects across all log entries. It should be noted here that any rule with a required support count and confidence value is deemed relevant by the algorithm; however they might still not be interesting or useful to the users. The ARM process performs scans entire dataset for items that have strong co-occurrences. The term 'strong' is defined by the amount of support and confidence. According to an analysis of 61 interestingness measures on 110

different datasets [127], using a right support value with respect to a dataset is essential for producing high quality rules.

3.2.2 Establishing Support Value

Given the fact that ARM algorithm has no prior knowledge about the dataset and the output is based on frequent itemsets, there is a possibility of extracting meaningless rules that do not add value to the security knowledge. For this reason, choosing a right support for ARM algorithm is critical for generating rules of better quality and quantity. In traditional ARM, the algorithm takes a single support value and outputs each rule with that support value or above. For example, if the input support is 0.5, all rules having support between 0.5-1 will be displayed.

Events triggered due to the security activities may be lesser in number as compared to the routine events. This is because system administrators do not often repeat the same security actions on a frequent basis. So, we can form an assumption that quality results be produced by lowering the support value. But, if the support value is kept at minimal to identify low frequency activities, it would allow more routine events to correlate, hence generating large amount of meaningless rules. Due to these issues, there is a 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 has different properties and specifying appropriate support thresholds without the knowledge of dataset can be difficult, we have devised an automated mechanism to determine the SR. The implemented ARM algorithm takes both minimum ($minsup$) and maximum ($maxsup$) support values and finds all those rules having $minsup \leqslant \text{support} \geqslant maxsup$. 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 [128] within the limits of SR.

The SR is calculated based on the object frequency distribution (OFD) of events types. The OFD is a set of positive numbers, where each element describes the number of times a unique object has appeared in a given event log dataset. Two other techniques

of a similar nature have also been developed [129, 130]; however, they do not calculate support values from the dataset but rather rely on the confidence to define threshold and filter interesting rules. The first approach is not capable of finding those interesting rules, which are composed of more than two items. This approach is unsuitable for events correlation as more than two events can exist in an associative relationship. The second approach generates all possible association rules, regardless of support and confidence thresholds, and then use transitive set property and manual examination to filter interesting patterns [131]. This approach too is not reasonable as it would require huge computing resources (or manual effort) for large-scale datasets. So in this research, the ARM algorithm employs frequency of objects to estimate the SR. The formulae used to determine the minimum and maximum support values from an OFD are shown in Equations 3.4a and 3.4b. It should be noticed here that the algorithm uses frequency of objects rather than frequency of events for SR calculation as the rules are mined from object-based model, not directly from event log dataset.

$$SR = \begin{cases} \frac{F_{min}}{F_{tot}} \quad to \quad \frac{F_{max}}{F_{tot}} & \text{if OFD is normal} \quad (3.4a) \\ \\ \frac{F_{avg}}{F_{tot}} \quad to \quad \frac{F_{max}}{F_{tot}} & \text{if OFD is not normal} \quad (3.4b) \end{cases}$$

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 any normal distribution, the data points are in symmetrical order and if the size of distribution is relatively small, there will not be a substantial difference between the minimum and maximum elements [132]. So 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 3.4a 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 is necessary during the calculation of SR. This can be performed using a normality test, which will allow the ARM algorithm to include interesting and useful rules, meanwhile, preventing the extraction of irrelevant rules.

3.2.2.1 Normality Test

Many methods are available to determine whether a given distribution is normal, such as Shapiro–Wilk (SW) and Two-Sample Kolmogorov–Smirnov (TSKS) tests. The distribution size can grow 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 [133]. Recent comparisons [134, 135] show that the TSKS test is an effective method among others and is suitable for large sample sizes. The TSKS test takes two one-dimensional distributions and decides if they significantly differ from each other. The TSKS is a non-parametric test [136], which means it does not make any assumptions about the distribution and quantifies a distance between the empirical distribution functions of samples. The process of normality test is provided in the following:

Step 1 (Generate a random normal distribution). The first step in the TSKS 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 in [137] that either takes a pair of standard deviation and mean values or first and last elements of the OFD to generate a reference normal distribution (RND). The RND has similar range of values as of OFD, which helps in increasing the accuracy of normality test.

Step 2 (Determine empirical distribution functions). The next step of TSKS test is to determine the empirical distribution function (EDF) [138] of both OFD and RND. The EDF assigns $\frac{1}{n}$ probability to n elements and outputs a discrete distribution. This uses the formula provided in Equation 3.5.

$$EDF_n(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{x_i \leq t} \tag{3.5}$$

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.

Step 3 (Test the hypothesis). The next step is to test the empirical distributions of RND (EDF_{OFD}) and OFD (EDF_{RND}) under the hypothesis that both 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 refuted, then the OFD is not normal. This is determined by finding the maximum set of distances (or differences) between the items of EDF_{OFD} and EDF_{RND} using Equation 3.6:

$$D = \sup |EDF_{OFD} - EDF_{RND}| \tag{3.6}$$

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 3.7.

$$D > 1.36\sqrt{\frac{n+m}{nm}} \quad \text{(critical value)} \tag{3.7}$$

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 [139].

Step 4 (Step-4: Repeat the test for better accuracy). This process is repeated with several, 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 TSKS tests, which consequently provisions better support values for correlation mining.

Consider an OFD shown in Equation 3.8. 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 of OFD, EDF_{OFD_1}, is presented in Equation 3.9.

$$OFD_1 = \{1, 2, 3, 4, 14, 22, 31, 59, 60, 68, 91, 144\} \tag{3.8}$$

$$EDF_{OFD_1} = \{0.08, 0.17, 0.25, 0.33, 0.42, 0.5, 0.58, 0.67, 0.75, 0.83, 0.92, 1.00\} \tag{3.9}$$

Now consider the RND shown in Equation 3.10. It is generated using the minimum (1) and maximum (144) values of OFD_1. The total number of objects in RND_1 is 12, minimum value is 9.80, maximum value is 137.35, average is 71.09 and sum of all elements is 853.09. The empirical distribution of RND, EDF_{RND_1}, is presented in Equation 3.11.

$$RND_1 = \{9.80, 40.60, 43.03, 57.11, 60.50, 73.42, 74.15, 79.30, 82.98, 90.93, 103.92, 137.35\} \tag{3.10}$$

$$EDF_{RND_1} = \{0.08, 0.17, 0.25, 0.33, 0.42, 0.5, 0.58, 0.67, 0.75, 0.83, 0.92, 1.00\} \tag{3.11}$$

After applying Equations 3.6 and 3.7 on EDF_{OFD_1} and EDF_{RND_1}, the value of D is found to be 0, which is not greater than the critical value 0.54. Hence, the hypothesis is proven false. After repeating this process for 12 times (as there are 12 elements), the dominant output still comes out as false. This means OFD_1 has failed the TSKS test and is considered as not normal. The Equation 3.4b will used to define a range of support values for OFD_1. The minimum support value would be $\frac{41.58}{499} = 0.08$, whereas the maximum support would be $\frac{144}{499} = 0.29$. Therefore, the ARM algorithm will mine all those (object-based) association rules, whose support is between 0.08-0.29. At this stage, correlations amongst event objects have been discovered and extracted, and the next stage is to translate these relationships to determine connections among event types. 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.

3.2.3 Event-Type Association Rules

As previously defined and repeated to increase readability, the 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 (*matched*), 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 but lower in quantity, whereas a lower value will generate a higher number of rules that might be of a lower quality.

For example, consider an object-based rule $X_0 \rightarrow Y_0$, where $X_0 = \{a, b, c\}$ and $Y_0 = \{a, b, c, d, e, g\}$ and two event log entries $E_0 = \{a, b, c, d, e, f\}$ and $E_1 = \{a, b, c, d\}$. All elements of X_0 are fully matched with E_0, but X_0 is missing 3 elements that are present in E_0. Hence the similarity amount will be $\frac{3}{(3+3+0)} = 0.167$. Again, all elements of X_0 are fully matched with E_1 as well, but X_0 is missing 1 element that is present in E_1. Hence the similarity amount will be $\frac{3}{(3+1+0)} = 0.75$. On the other hand, the elements of Y_0 are not a full match with E_0, and the similarity amount is $\frac{5}{(5+1+1)} = 0.71$. Likewise, the elements of Y_0 are not a full match with E_1, and the similarity amount is $\frac{4}{(4+0+2)} = 0.67$. Therefore, the rule $X_0 \rightarrow Y_0$ will become $E_1 \rightarrow E_0$ as X_0 and E_1 have 75% similarity, whilst, the Y_0 and E_0 have 71% similarity.

In this manner, the proposed solution processes LHS and RHS of all object-based rules, which are then replaced with corresponding event types (numeric identifiers). At this point, there is a possibility of same event type occurring in both LHS and RHS. In that case, the event type having the highest similarity with rule objects remains, whilst the other one is removed. If both events have equal similarity to their respective LHS and RHS of a rule, both events will be excluded from the event-based rule to remove the error and prevent any ambiguity as well.

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 currently 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 $\{1234, 4567\} - \{5678\}$.

3.3 Sequences Of Event Relationships

This section presents the process of identifying the order of events in event-based rules by employing a temporal metric. Following on, individual rules are formulated into sequences of events which are later validated as well.

3.3.1 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 time-stamp with every entry. The proposed algorithm utilises a temporal metric for ordering of events, which is based on the time-stamp of entries. In other words, the ordering of events in all rules is determined based on when the events were generated and logged. According to a survey [140], applying temporal metric to several algorithms has improved the quality and efficiency of results in different domains, such as statistics, machine learning and databases. The survey mainly targeted those algorithms that take sequential data streams and discover interesting patterns. This research presents a new Algorithm 1, which takes the event-based rules and the event log dataset as input and assigns a direction to each event relationship, based on the temporal accuracy values.

The algorithm 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 behind this approach of finding all subset combinations is to determine if there exists a temporal link between each pair of correlated events. Processing each subset combination of all rules is a computationally expensive task, especially if there are large number of event-based rules. Several techniques do exist that can decrease the number of rules, and as a result decrease the time and resource consumption of overall process, such as iterative rule pruning based

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

procedure TEMPORAL-ASSOCIATION-RELATIONSHIP
  Initialise $C \leftarrow \emptyset$
  for all $r_i \in R$ **do**
    $(r_x, r_y) \leftarrow r_i$
    for all $EventType_x, EventType_y \in r_x, r_y$ **do**
      $PosE_x \leftarrow$ **GetIndicies**$(EventType_x, D)$
      $PosE_y \leftarrow$ **GetIndicies**$(EventType_y, D)$
      $t_f \leftarrow$ **Count**$(\forall x \in PosE_x < (\forall y \in PosE_y))$
      $t_s \leftarrow$ **Count**$(\forall y \in PosE_y < (\forall x \in PosE_x))$
      Initialise $TAA \leftarrow 0$
      Initialise $direction \leftarrow 0$
      if $t_f > t_s$ **then**
        $TAA \leftarrow t_f/(t_f + t_s) \times 100$
        $direction \leftarrow 1$ ▷ means $X \rightarrow Y$
      else if $t_f < t_s$ **then**
        $TAA \leftarrow t_s/(t_f + t_s) \times 100$
        $direction \leftarrow -1$ ▷ means $Y \rightarrow X$
      end if
      if $TAA > 50$ **and** $direction$ is 1 **then**
        $C.Add((EventType_x, EventType_y), TAA)$
      end if
      if $TAA > 50$ **and** $direction$ is -1 **then**
        $C.Add((EventType_y, EventType_x), TAA)$
      end if
    end for
  end for
end procedure

on a user-defined criteria [141] and rule pruning based on a probabilistic classifier [142]. However, inspecting each subset combination reduces the risk of missing any interesting rule that can be beneficial for the user, hence improving the quality of overall process. In this research, we trade-off time and resource consumption with quality, therefore generating high accuracy rules.

The next step is to determine the temporal-association accuracy (TAA) of all subset combinations from each event-based rule. This will facilitate the conversion of correlation

rules into temporal-association relationships. The TAA value depicts the number of times a certain relationship was found accurate according to the given dataset, i.e. correct event ordering based on the temporal sequencing of events. So if an event was logged first with respect to time, it will appear first in the event ordering as well. It is also demonstrated by Pearl [143] 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. Hence to eliminate or at-least reduce this issue, we have used the event-generation rather than event-written timestamp. Event-generation timestamp denotes the time at which this event was triggered and submitted to the logging service, whereas, event-written is time at which this event was received and written to the log file[1].

To determine TAA value, the indices of event type from LHS (line 6), as well as RHS (line 7) of a subset are gathered 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 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 a minimum of 50% TAA value due to the two-way comparison. 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.

[1]https://docs.microsoft.com/en-us/windows/win32/api/winnt/ns-winnt-eventlogrecord

A threshold value of above 50% TAA is chosen to select temporal-association rules as shown in lines 19 and 22. Due to these conditions, only those rules will be selected for further processing, where LHS event occurred before RHS more than half of the times, 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 applied 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 depends on the user 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 event log dataset containing 200 entries, and the proposed solution outputs $(1234, 4567 - 5678)$ event-based rule (as explained in Section 3.2.3). The first step is to divide the rule into two subsets, i.e. $(1234 - 5678)$ 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 $(1234 \rightarrow 5678)$ 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 $4567 \rightarrow 5678$ with TAA value of 70%.

3.3.2 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 construct sequences of temporal-association relationships. The temporal-association sequences demonstrate 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 groups of subsets that have common RHS event type. After that, the solution combines the LHS event types of each group. The reason behind this step is to identify and connect all similar events (taken from the LHS of subsets) into separate groups that were performed to achieve corresponding common goals (identical RHS of all subsets). Although at this point, each group is an unordered set of events that represents one or more particular actions to conduct a single security-related activity or configuration. 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. Again, each member $g \in G$ will have one or more event types on LHS linking a single event type on RHS.

The next step is to create an ordered set of events within every $g \in G$, so that each group of correlated events can be formulated into a chain to depict the correct sequence of events. The first step is to identify those two event entries, which have a maximum time difference between them. Such two events will be used as the starting and ending events of action(s) that were performed on the underlying system. Similarly, determine the second to last event based on the time that had happened before the ending event. Repeat the process until all events are covered. This process will organise the event entries within g with respect to time; hence, creating the initial set of temporal-association rules (or sequences of events). The next step is to validate the extracted rules. This process calculates a TAA value of each pair again in the sequence. After that, it measures a average/mean of all TAA values, which is considered a collective TAA value of the sequence. The final TAA value of each sequence will be at-least 50%. The sequences having high TAA values that are more closer to 100% are deemed as more accurate and fine-grained for the user.

Continuing the same example from Section 3.3.1, $g = (1234, 4567 \rightarrow 5678)$, 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), which would be considered as a common goal. Referring back to Figure 3.2, 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 $(4567 \rightarrow 1234 \rightarrow 5678)$. The TAA values will be calculated between the following pairs: $(4567 \rightarrow 1234)$ and $(1234 \rightarrow 5678)$. Assuming the values were found to be 70% and 80% respectively, the collective TAA of the sequence will be $(70 + 80)/2 = 75\%$. Hence the chains of events are formulated using frequent pattern observation of events (ARM algorithm) 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 [144], 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. For example, before it was known that smoking causes cancer, researchers assumed that the symptoms like yellow teeth, hair loss and pale skin cause cancer. These symptoms are in fact effects of smoking, which also leads towards the cancer. So first of all, if two events, say 4567 and 1234, are not in an associative relationship, they cannot form a cause and effect relationship. But if both events are directly associated and the correlation is falsified when a third event 5678 is introduced, 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 the accuracy of extracted knowledge. This motivates the need to find the strength of causality in the extracted sequences of events.

3.4 Determining 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 defines what (one or more) activities in the underlying machine led to

a particular activity. To put it differently, causality refers to the relationship of two or more events, such that they inform of a finite sequence of actions, which were performed one after another by an expert in order to improve the security of a vulnerable machine. The causal relationship can be both static and dynamic. The static causality means that 'Event A must always occur for Event B to happen', whilst, the dynamic causality is 'Event A caused Event B' depending upon the circumstances, system environment and runtime conditions in which the events were generated. As the extracted temporal-association sequences (process described in Section 3.3.2) are not universal, but subjective to the event log dataset, the causal relationships in our case are dynamic.

As discussed in Section 2.3.4, the existing algorithms are based on either Bayesian or Constraint-based approaches. The Bayesian approach defines the cause and effect relationship in terms of probability, whereas the Constraint-based approach establishes it conclusively, i.e. either conditionally dependent or independent. Moreover, the Constraint-based approach is considered as high in performance and efficiency for large-scale datasets. Due to these reasons and benefits, this research uses constraint-based approach to discover causal relationships among the correlated events. We first create a directed acyclic graph from the given set of temporal-associative relationships as it is a requirement for causal inference algorithms. This involves the careful elimination of relationships, which do not satisfy the criteria. Following on, an algorithm is used to determine the causality or causal rank of each relationship. In the output, the rules with higher causal rank are deemed as more reliable.

3.4.1 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. The set V contains n number of elements and corresponds to all unique event types, whereas the set E represents their associations. An edge between event type i and j is called directed if the condition $(i \rightarrow j) \in E$ but $(j \rightarrow i) \notin E$ is satisfied. A directed acyclic graph (DAG) is a type of graph, where every edge is directed, i.e. does not have conflicting edges and cycles. It is important to create a DAG for finding causality 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 the implementation easier for further processing. The next step is to remove duplicate and conflicting subset rules based on temporal-association accuracy (TAA) value. For example, if $E_1 = (i \rightarrow j)$, $E_2 = (i \rightarrow j)$ and $E_3 = (j \rightarrow i)$, and all three edges exist in G at the same time, find the TAA value of each edge to prioritise them. First, the E_1 and E_2 are duplicate edges. If $TAA_{E1} \geq TAA_{E2}$, then remove E_2, otherwise E_1. Second, the E_1/E_2 and E_3 are conflicting edges and the one with the lower TAA value will be removed. These steps are necessary to eliminate ambiguities from the final result and is also a condition to build DAG. It should be noticed here that the number of event types will remain the same overall, and only the associations will be removed.

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 cycles are important to remove for an automated solution as it would prevent continuous repetition/duplication of rules in the final results. The proposed solution uses topological sorting using Kahn's Algorithm to detect cycles [145]. The overall time complexity of Kahn's algorithm is $O(V + E)$. Kahn's algorithm first determines the number of incoming (in-degree) and outgoing (out-degree) edges for each vertex. The process is started by traversing the neighbours of a vertex, whose in-degree is initially zero, i.e. no incoming edge. The algorithm maintains an ordered list of vertices, whose in-degree is or becomes zero during the process. 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 list. 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 for cycles again. This process is continued until the graph becomes completely acyclic.

Consider the event temporal-association rules and their TAA values (in brackets) presented in the first column of Table 3.1. The event types are represented in terms of A,

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

B, C and so on. There are nine rules in total and the table demonstrates the three-step 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 as shown in second column. In the second stage, the rule $(B \rightarrow E)$ occurred two times (at number 4 and 8), and one of them is removed as shown in third column. 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 cycle-free rules are shown in the last column of Table 3.1, which represents the conversion of temporal-association rules into a DAG. This DAG will be now used as input to the causal inference algorithm to determine the strength of causal relationship for each edge.

3.4.2 Inferring Causal Rank

Several models and algorithms are available to perform causality analysis in a graph. Most of them are the variants of following two algorithms: Peter-Clark algorithm (PC) [146] and the subsequent improved version named Fast Causal Inference (FCI) [147]. For this research, the FCI algorithm is applied as it allows hidden variables, is scalable to high-dimensional data and has better accuracy. Moreover, the first half 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 [148] that search for causal constraints among all vertices and edges of a DAG, and output a unified model of causality.

3.4.2.1 PC Algorithm

The PC algorithm is a fundamental and powerful structure-learning algorithm for DAGs. It evaluates patterns in the DAG to determine whether they are consistent with hypothetical DAGs representing possible causal structures. The PC algorithm has also been used in several real time applications, such as analysing orthopaedic implant data for hip replacement signal detection [149] and discovering relationships among the daily index values of the four prominent teleconnection patterns in the atmosphere [150]. An efficient implementation of the PC algorithm for high-dimensional data is presented by Kalisch and Buhlmann [151]. 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.

Step 1 (Create DAG skeleton). 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 the $X - Y - Z$ skeleton. Processing the skeleton of a DAG creates an additional layer of validation and assertion of previously acquired relationships as the directions are nullified for this stage and it will infer the causality without any bias. Following on, the algorithm creates an adjacency matrix of the skeleton and performs a conditional independence (CI) test between each adjacent pair of vertices (event types) 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 and represents a complete graph. The CI test is based on the Causal Markov Condition, i.e. given parents, the event is conditionally independent of all non-descendants or the event is conditionally dependent on its parents (direct causes). In other words, the occurrence of event is independent of every other event in the DAG, except its effects.

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

Step 2 (Perform conditional independence tests). The PC algorithm employs a d-separation technique to perform CI test [152]. 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. In other words, the events X and Z are d-separated if all paths that join them are *inactive* relative to other events, or simply, there is no *active* path between them [153]. 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 connected events in a sub-path of graph are not related to each other, and hence either the whole sequence of events is either incorrect or it is missing some crucial events that are necessary to define events sequence (i.e. a complete expert security action). Hence, the event relationships where the Causal Markov Condition is satisfied are the least reliable in terms of causality as they are not dependent on each other. 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.

Step 3 (Orient v-structures). After performing the CI tests on the adjacency matrix, the PC algorithm applies orientation rules to the skeleton of the DAG. The orientation is a process of assigning directions to the two ends of undirected edges, forming an equivalence class of the DAG. The first step is to consider each triplet of vertices (X, Y and Z), such that the pairs (X, Y) and (Y, Z) are each adjacent in the skeleton but (X, Z) is not, based on Y being empty or not. Consider the first path presented in Figure 3.4 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 [154]. This path will be assigned a low priority with respect to causal strength 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, Y) and (Z, 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 and 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' and will be applied to all triplets of vertices in the DAG.

Step 4 (Orient other remaining edges). After identifying all v-structures, the PC algorithm orients the remaining edges in a way that does not introduce 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. This can be determined by examining the remaining paths (2, 3 and 4) of Figure 3.4:

- *Empty conditioning set* – The event Y is unobserved. All of the following paths are active, as Y is the non-collider, and allows X and Z to be connected without any condition:
  - In the second path, the event X is an indirect cause of event Z;
  - In the third path, Z is an indirect cause of X; and
  - In the fourth path, Y causes both X and Z;
- *Non-empty conditioning set* – The event Y is known. All of the following paths are rendered inactive, as Y is the non-collider and blocking a connection between X and Z:
  - In the second path, the event X can cause Z, if and only if event Y occurs;
  - In the third path, the event Z can cause X, if Y occurs; and
  - In the fourth path, only event Y can trigger X and Z.

All inactive paths are assigned a low rank in terms of causality due to d-separation. Table 3.2 provides a summary of finding active and inactive paths based on whether the conditioning set is empty or non-empty.

TABLE 3.2: Determining the active and inactive paths

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

Based on the above discussion, the PC algorithm processes the skeleton of DAG and assigns appropriate directions. As all graphs are represented by an adjacency matrix, it allows the algorithm to be more efficient for the high edge-density graphs. However, a drawback of PC algorithm is that it can propagate errors if a mistake is made in the earlier stages. Furthermore, the presence of hidden variables limits the algorithm to consider only subsets of the adjacent vertices 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 edges, due to lack of fully accurate CI tests, that can be eliminated (or assigned low priority in our case). Such edges can be determined by the FCI algorithm, which is described in Section 3.4.2.2.

3.4.2.2 FCI Algorithm

The FCI algorithm is an extension and more generalised version of the PC algorithm that takes the output of PC algorithm and applies additional CI tests [155] to learn the structure of graph. The PC algorithm alone might produce incomplete results due to the presence of hidden variables in the DAG. A hidden variable, sometimes also referred as confounding or latent, is an unrelated or rather meaningless variable that correlates with both dependent and independent variables, i.e. it is an unobserved hidden common cause [156]. In the current solution's context, a true causal structure among events cannot be inferred without exposing all hidden variables as there might be other connections influencing the relationship. The FCI algorithm includes additional orientation rules to discover and represent such relationships, and consequently provide complete and sound output.

To perform the CI tests between the items of an edge (e.g. X and Z), the FCI algorithm applies two additional 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 sequence of distinct adjacent vertices (path) ϕ between V_i and X or V_i and Z, such that either (a) ϕ contains a collider; or (b) ϕ is not marked as

a non-collider and $X - \phi - Z$ form a triangle. A triangle is a set of three vertices all adjacent to one another. 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 hidden 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 [157]. Hence the FCI algorithm is capable of providing complete and sound set of causal relationships in a reasonable amount of time.

The FCI algorithm uses a type of mixed graph, called Maximal Ancestral Graphs (MAGs), for representing the collection of all conditionally independent relationships between the observed variables (depicting events in our case). A mixed graph is a vertex-edge graph that can have three possible edges: directed ($\rightarrow$), bi-directional ($\leftrightarrow$) and undirected ($-$). It can have at most one edge between any two vertices. A mixed graph is ancestral if (a) there are no directed cycles and (b) no such vertex X, which is an ancestor of any of its parents or 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 [158]. The m-separation is a graphical criterion to encode conditionally independent relationships, and can also be defined as an equivalent of d-separation in the DAG. It measures the graphical disconnectedness in MAGs. To define m-separation, first consider the definition of its opposite 'm-connecting'. A path ϕ between two distinct X and Z vertices of the ancestral graph is m-connecting (or active) relative to a (possibly empty) set of vertices Y, where $(X, Z) \notin Y$, if (a) every non-collider on ϕ is not a member of Y and (b) every collider on ϕ 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.

Another property of MAGs is the representation of marginal independence (MI) models of DAGs. Consider the following scenario: given any DAG containing a set of both observed and latent variables, there is a MAG of observed variables, such that X and Z are d-separated by Y, if and only if X and Z are m-separated by Y. So, MI means that only two events (X and Z) are considered independent 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 property of MAGs, several MAGs can describe the same set of conditionally independent relations, which were identified by (a) d-separation and

(b) Possible-D-SEP with respect to latent variables, and graphically encoded by the m-separation [159]. Such MAGs are considered to be Markov equivalent [160]. The multiple MAGs generated by the FCI algorithm can have same adjacent vertices and (usually) common edge orientations [161], which is why it is not fully testable with observational data. This raises the need for a single, unified graph that is capable of representing all MAGs simultaneously. The unified graph is the output of FCI algorithm and called Partial Ancestral Graph (PAG).

A PAG describes causal features common to every MAG in the Markov equivalence class. It has been proven [162, 163] that the output of FCI algorithm is maximally informative due to PAG being capable of holistically representing the directed, undirected, partially directed and double-headed edges. A PAG can have three end marks for edges: arrowhead ($>$), tail ($-$) and circle (o). It can form four kinds of directed edges: $\rightarrow$, $o\!-\!o$, $o\!\rightarrow$ and $\leftrightarrow$. The PAG has the same set of adjacencies as of MAGs and represents all 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 output edges [164, 165] with respect to proposed solution's context. Each edge is assigned a specific causal rank between 0 and 4, where 0 is the lowest and 4 is highest causal strength:

1. $X \rightarrow Z$: both X and Z are conditionally dependent, and X is a direct cause of Z. This is the strongest causal 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 sure about the 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 caused Z or Z caused X. This uncertainty occurred because both of these relationships were found in different MAGs. We have assigned this a rank 2;

4. $X \leftrightarrow Z$: the bi-direction indicates that this edge was influenced by one or more hidden variables, and that lead to X and Z having a common 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; and

5. All other associations, where the FCI algorithm assigned either (a) undirected edges to show lack of sufficient knowledge to form causal structures; or (b) no edge at all to show the conditional independence between vertices, are given rank 0. The reason being FCI could not establish any evidence of causal connection.

Our solution uses the R package called, *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 [166]. 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 the DAG are assigned corresponding causality ranks based on the aforementioned criteria. The implementation of FCI in proposed solution is somewhat similar to Linear Non-Gaussian Acyclic Model (LiNGAM) algorithm [167] in assigning scores to edge directions. The LiNGAM algorithm uses a statistical method known as independent component analysis to develop causal structures and prune edges in the DAG. However, instead of removing the event temporal-association relationships, the solution assigns a causal rank (priority) to signify the strength of causal relationship. The conversion of temporal-association into causal rules, referred as temporal-association-causal rules from this point, provides an additional layer of confidence in terms of reliability and accuracy [168].

Continuing the example from Table 3.1, the extracted DAG is shown in Figure 3.5 (L), whereas a resultant PAG along with an assignment of causal rank on each edge is shown in Figure 3.5 (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 relationships. Each ($o\rightarrow$) 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 decides 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

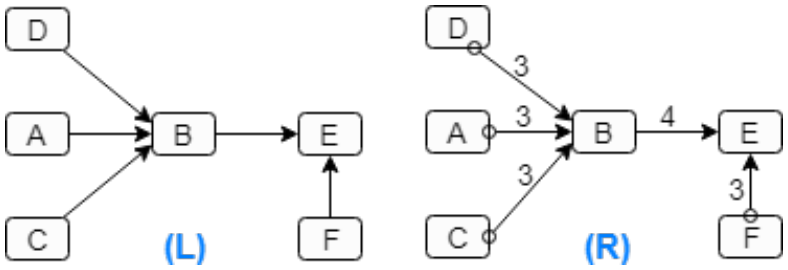

FIGURE 3.5: (L) DAG created from the temporal-association rules given in Table 3.1 and (R) a PAG created from DAG where each edge is assigned a causal rank, using the FCI algorithm.

the (user) actions and (administrative) responses in terms of security as the actions have no cause events, unlike responses.

3.5 Storing Rules

Now the temporal-association-causal rules have been established and will be stored in a simple database. The schema of the database is shown in Figure 3.6. The database contains two tables; (a) 'Events' to store event information and (b) 'Rules' to store the rules. The 'Events' table contains a list event types, their descriptions and corresponding objects. This information is used to elaborate the rules that makes it easier for the non-experts user to understand. All event data is anonymised, such that it does not contain any sensitive or personal information. In the 'Rules' table, each row represents a rule, which 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. These 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 numeric value $(50-100)$ is calculated by Algorithm 1; and
- *Causal Rank* – This numeric value $(0-4)$ is calculated by FCI algorithm as described in Section 3.4.2.

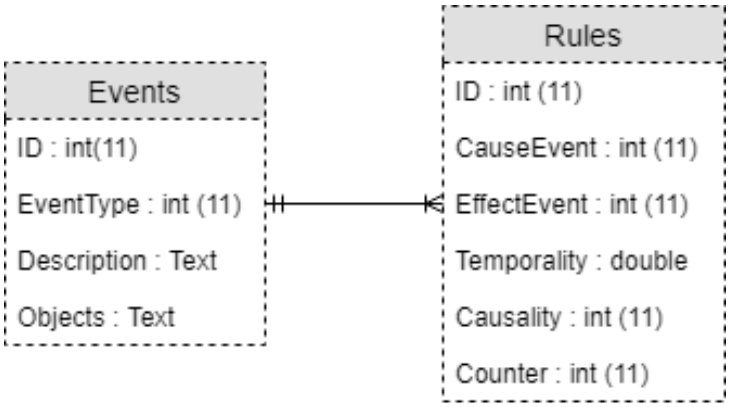

FIGURE 3.6: Database schema to store temporal-association-causal rules along with the anonymised event type information

The rules acquired from all event log datasets are stored in a same table. 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. When new set of TAC rules are extracted from an unseen dataset, the new and existing rules collectively go through the process of inferring causal ranks again. Although this is a resource intensive task, it is necessary to always keep updated rules and values, which are also 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.

3.6 Chapter Summary

In this chapter, a novel mechanism of acquiring knowledge from event log entries of an operating system has been described. The knowledge is extracted in the form of temporal-association-causal (TAC) rules, which are essentially a sequence or pattern of connected events. Each sequence depicts specific action(s) performed for security configuration or to remediate a security issue. All TAC rules are stored in the database along with the relevant information for application.

The process starts by removing the routine entries from given event log dataset, and then creates an object-based model of the remaining entries. The object-based model is used as an input for the association rule mining (ARM) algorithm to learn correlation rules among events. The support for the ARM algorithm is calculated automatically using the same object-based model. The correlation rules are then transformed into

sequences of temporally-associated rules using the timestamps of events. At the end, causal inference algorithm is applied to the temporal-association rules that assigns a rank to each relationship in terms of causal strength. In a nutshell, the event relationships are defined by associations, ordered by time and consolidated by causation.

Most of the existing solutions for discovering event relationships use manual approach, where human experts are involved at certain stage in producing the correlation rules. Such techniques are not feasible for the complex and large-scale datasets as it would require massive amounts of time and effort in processing and continuously updating the rules. The proposed solution is fully automated and can process any number of event log entries. Furthermore, the proposed solution applies dynamic causal inference for finding cause and effect between events to produce more reliable results.

Chapter 4

Automated Planning

Utilising the extracted knowledge in the form of temporal-association-causal (TAC) rules on a previously unseen machine is not feasible as algorithmic support is required. The TAC rules present a collective set of expert security 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 required and useful for a particular security issue or missing configuration. Another important aspect to consider is the order of acquired actions, which might not be in the same order that would be applied on an unseen machine. Therefore, a precise, intelligent and clear decision-making process is needed regarding which actions to select for any given machine. The problem of selecting appropriate actions for different settings is a deliberation problem, which is traditionally performed manually by human experts. However, the manual technique demands knowledge, can be error-prone and requires a significant amount of time, effort and resources. So we overcome this challenge by employing automated deliberation techniques in the form of Automated Planning (AP). This is particularly an attractive solution for security assessment and configuration as previous applications in the similar areas have shown that it can potentially provide a method of eliminating representation and planning complexities along with finding the most efficient solution and reduce manual effort. Hence, this chapter describes the final stage of the research, which 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.

4.1 Classical Planning

The origin of AP can be traced back to a General Problem Solver (GPS) developed in 1963 [169]. The GPS takes a problem and divides it into sub-goals before reaching the final goal. It uses means-ends analysis to identify and reduce the differences between the current state and the goal state. Later on, another approach for planning was introduced, called as situation calculus [170]. This approach provides a theoretical framework to represent actions with clear semantics by 'reify' situations. The meaning of reify is to treat something abstract as an object. This introduced the concept of functions, planning strategy, actions and logic along with the predicates containing variables and constants to describe a situation or state of the world. Classical planning is the name that is given to such earlier methods of planning.

In Artificial Intelligence (AI), problem solving is defined as an organised search through a range of predefined actions to achieve particular goal from a present condition. It also involves synthesising domain models and domain-specific constraints. The AP is a type of general problem solving and has been a widely researched discipline of AI for nearly six decades. The problem solving is performed by planning algorithms, also referred as planners, which generally take (1) domain model that describes the knowledge in the form of permitted actions and (2) a stand-alone problem instance that needs to be solved as input and generate plan of actions to solve the given problem [1]. The planners mainly deal with the order in which the activities should be performed, i.e. 'what has to be done'. Formally, the planners are responsible for the deliberation process, which consists of reasoning with a knowledge-base of actions against a given problem and explicitly arranges the output in a particular sequence to yield a sensible plan solution. A complete plan solution starts from an initial state of the world and depicts all legal actions to achieve desired goal state of the world. Notice that having sufficient and correct knowledge, which is TAC rules in our case, is integral to the efficiency of planner to produce useful plans and enable the autonomic properties.

4.1.1 Conceptual Model Of Classical Planning

To explain the fundamental idea of autonomous planning, a full conceptual model is shown in Figure 4.1. It consists of the following three components:

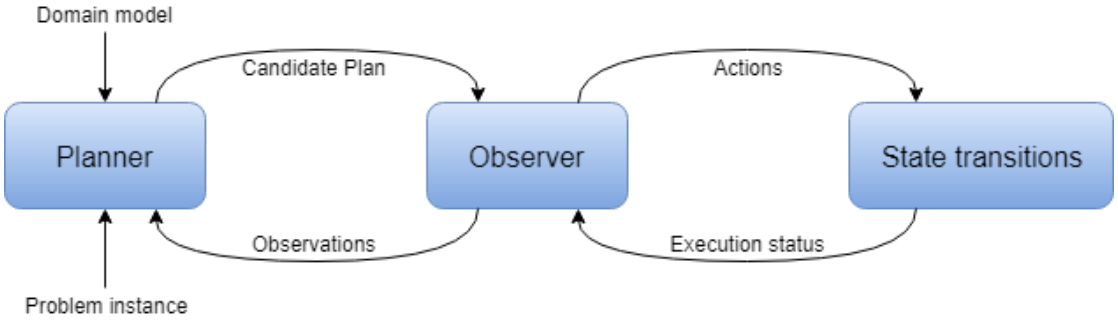

FIGURE 4.1: Concept of AI planning (inspired from [1])

- *Planner* – produces a plan (ordered set of actions) for a specified problem instance using the knowledge encoded in a domain action model;
- *Observer* – examines the current state of the system and chooses an action from the plan based on a state-transition function; and
- *State transitions* – a system that progressively performs the state transitions according to the actions received from the observer.

4.1.2 Formalisation

The formalisation is based on the state transitions system, which is the dynamic modelling of transitions from initial to goal states. A dynamic model describes that how one state transforms into another state. The state transitions system employed by the AP algorithms is a 3-tuple $\sum = (S, A, \gamma)$, where:

1. $S = (s_1, s_2, \dots)$ is a finite set of states;
2. $A = (a_1, a_2, \dots)$ is a finite set of actions; and
3. $\gamma : S \times A \to 2^s$ is a state-transition function.

A state is the factored representation of a 'state of the world' containing the combined domain knowledge, which is gathered through agents in manual, automated or both ways. Actions are transitions that are determined by the planer. If $a \in A$ is an action and $\gamma(s, a) \neq \emptyset$, the action a is valid for state s.

A solution P is a sequence of actions $(a_1, a_2, \dots, a_k)$ respective to a sequence of state transitions $(s_1, s_2, \dots, s_k)$, such that $s_1 = \gamma(s_0, a_1), \ \dots, s_k = \gamma(s_{k-1}, a_k)$, and s_k is the

desired goal state. The model's configuration is represented by a set of first-order predicates that uses quantified variables over non-logical objects. The predicates 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 $eff-$ shows the action's positive and negative effects.

4.1.3 General Assumptions

Following list provides the common assumptions that are made while modelling the knowledge. Such assumptions limits the complexity through abstracting the problem, as well as defines the boundaries of the modelling process:

1. The system $\sum$ is fully observable. The domain model contains complete knowledge regarding security assessments and configuration;
2. The system $\sum$ is deterministic, which means applying an action to a state always outputs a single successor state [171]. The domain model and problem instances are already known and there is no uncertainty involved, i.e. the output plan is predictable and will be found if it exists;
3. The system $\sum$ contains finite number of actions, where each action represents a relationship between two events;
4. The system $\sum$ is static. It is fully controlled by the planner;
5. The goals are restricted and explicitly specified. The objective is to identify any sequence of state transitions that reaches the goal state(s);
6. A solution plan contains a linearly ordered finite sequence of actions;
7. A domain-independent planning approach is used, which enables general use of constraints and resources to represent actions. This also eliminates future maintenance challenges and allows the reusability of domains models across related areas, including the development of new planning techniques;
8. The actions have no duration and there is no explicit notion of time; and

9. Any change in $\sum$ during planning will not effect the planning process, i.e. offline planning.

4.2 Process Of Knowledge Representation And Modelling

Knowledge Representation (KR) is a method of describing expert knowledge and is usually considered as a main issue of AP. The KR encodes the knowledge using symbols that can be processed and manipulated by a software tool to take intelligent decisions. It should include consistent and precise description of notions, facts, constraints and actions of the world. A major aspect of AP process involves converting expert knowledge into a formal domain model, such that it is valid, aligned with machine representation formalism to solve complex problems and can be utilised by a large variety of application domains. Another notable aspect is selecting a feasible representation language that can comprehend the complexity of planning problems at hand.

To define a well-modelled scenario, consider the two constants D and K, where D is a domain model and K is the real-world knowledge that D is representing [172]. The D consists of symbols that describe and map the attributes and operations of K. A simple example of K can be an administrative action, where a user creates a new password that includes both lowercase and uppercase letters. To represent this as D, we can use a symbol called, 'Password', along with corresponding arguments 'User-1', 'Lowercase' and 'Uppercase'. The statements (or actions) of D correspond to a true value in the real-world. So, if true in real-world, the D has 'password of User-1 contains lowercase and uppercase letters' action to reflect the situation. Thus, by using a mapping function, D represents a logical structure and an abstraction of K.

Modelling real-world scenarios can be a complex process, and correctness is of the essence to obtain a quality plan solution. Hence it is necessary to validate domain models against the knowledge. It is a challenging task as comparing domain model with real-world scenario is not possible, and the consistency/equivalency cannot be proved formally. However, several research studies [173, 174] have identified the following factors, which should be considered while creating a domain model:

1. *Accuracy* – It is an informal process that matches the features represented in domain model D with the real domain experts [175]. It checks the accuracy of

assertions in every state through comparison. It also involves checking consistency within D that can identify errors;

2. *Adequacy* – The domain D would be considered as adequate, if it is neither missing nor has any additional details with respect to real-world knowledge K;

3. *Completeness* – The domain D is considered as complete if it satisfies both accuracy and adequacy conditions. In addition, all real-world requirements should be present in D, such that every acceptable action plan to a related problem should be derivable from D; and

4. *Operationality* – There can be several domain models (D_1, D_2, $\cdots$) of any given knowledge that output quality plan solutions. However, some of the models would be more efficient in terms of computational resources in finding an acceptable plan. Such models would be considered more operational.

4.2.1 Modelling Process

Although there is no standard process to convert the knowledge into a formal domain model, but there are some general recommendations that can be used to perform systematic domain modelling process [176]. The first step is to conduct an early analysis to identify the sources and environmental characteristics of the knowledge. This can be done through the survey of similar applications and technical literature in that particular knowledge category. The survey will also help in collecting domain requirements. The next step is to choose a modelling language to represent the actions and their relationships along with the identification of state variables, types, constraints and preconditions/effects. After that, acquire the knowledge from available sources through either manual or automated mechanisms. Following on, encode the domain model (manually or automatically) to represent the knowledge using declarative descriptions. After designing the domain, it is important to perform the verification and validation checks to determine the model's accuracy and completeness. Further steps include continuous maintenance and improvement of the domain model according to the new additions in the knowledge.

In this research, the knowledge acquisition and domain modelling process is performed in an automated manner. The knowledge is present in the form of TAC rules among

other relevant information as described in previous chapter 2 and stored in the database. The knowledge modelling is performed by a software tool that takes the TAC rules and output a complete domain action model that can be utilised by supporting planners.

4.3 Representation And Utilisation Of Extracted Rules

A variety of knowledge representation languages has long been available to encode domain action models as described in previous Section 2.4.2.1. The choice of language partially depends on the requirements of the planning application itself along with some other factors, for example, is language commonly used, how many planner tools support the language, level of support for operational aspects of the domain model, availability of sufficient documentation etc. The language should be expressive, customisable and structured, so that it can capture and represent complex scenarios of the real-world. It should also contain a strong set of operator definitions that enables inference capability. Apart from these attributes, the language should have a clear syntax and semantics.

The PDDL version 2.1 has been chosen in this research to represent event relationships using the acquired temporal-association-causal (TAC) rules. It is a classical representation that allowed the use of typed state variables, numeric fluents and plan-metrics in our research. It also improves the efficiency of a planning system in terms of memory usage by decreasing the number of instances it needs to create [177]. The PDDL serves as a bridge between the knowledge acquisition phase and planning algorithm. It uses simple and intuitive constructs that are sufficient to express the problem. Since TAC rules have strict parameter, type and sequence requirements, failing to accurately fulfil those requirements in the domain model would result in plans that would not be beneficial for the users. This forces our PDDL representation of the TAC rules to be quite verbose. On top of that, we take advantage of the optimisation metric of planning algorithms that increases the quality and quantity of selected actions in an output plan.

4.3.1 System Definition

Choosing PDDL allows a categorical separation of elements belonging to domain and problem representations. This common format of representation has been widely extended and provides better knowledge distribution and more direct comparison of systems and approaches. The PDDL-based encoding of extracted rules also made it possible to choose an efficient AP algorithm out of many, therefore producing a quality plan of actions. The following sections provide the detailed explanation of encoding TAC rules that are stored in the database into a domain action model and the event log entries of a vulnerable machine into a problem instance. Specific to our research, a PDDL planning problem can be formally defined as a 5-tuple $P = (s, f, a, i, g)$, where:

1. s is the set of all possible predicates along with their parameters. Each predicate represents a unique event type and its relevant property/object names;
2. f is the set of all possible numeric fluents. A single fluent is used in our domain model to accumulate action costs;
3. a is the set of all possible actions that can be applied to a given problem. Each action represents a cause and effect relationship between two event types;
4. i is the set of predicates representing the initial state of a vulnerable machine. Each predicate is initialised with the respective event type property/object values;
5. g is the description of goal that should be achieved; and
6. P is the output plan that contains sequence of actions discovered from initial to goal state. Ideally, the plan should identify security issues and missing configurations of the vulnerable machine, and then propose solutions as well.

4.3.2 Domain Modelling

Consider an example of TAC rule presented in Table 4.1. This information is used to produce a domain action. It shows a cause and effect relationship between two events 4720 and 4673. The event type 4720 means that a new user account was created, whereas 4732 informs that the account was created by using administrative permissions. The object names and other information given in the rule are imperative for the planning

Rule property	**Value**
Cause event	*Type*: 4720 *Object names*: DisplayName, HomeDirectory, HomePath, PasswordLastSet, ProfilePath, SamAccountName, ScriptPath, SubjectDomainName, SubjectLogonId, SubjectUserName, SubjectUserSid, TargetDomainName, TargetSid, TargetUserName, UserWorkstations
Effect event	*Type*: 4673 *Object names*: ObjectServer, PrivilegeList, ProcessId, ProcessName, SubjectDomainName, SubjectLogonId, SubjectUserName, SubjectUserSid
Counter	3
Temporal-association accuracy	95
Causal rank	3

TABLE 4.1: An example of TAC rule for generating a PDDL domain action

algorithm to generate a quality plan of actions, which will consequently provide the non-experts with an actionable knowledge. The remaining section will use the example TAC rule to demonstrate the domain modelling process.

In this research, we encode the TAC rules and output a PDDL domain model file. A domain model consists of types of objects, predicates modelling facts, functions to handle numerical operations and actions. The objects and predicates are used to model the underlying system and an action represents an evaluation or configuration step to increase the security. A precondition depicts at what point the action is selectable with respect to a problem, and an effect models a change to the objects, predicates and functions, and is a step towards achieving desired goal. 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 through temporal metric.

4.3.2.1 Domain Name, Requirements And Types

The domain name is formatted as {EventRelations–<Number>}, e.g. EventRelations–1. The domain model uses 'typing' and 'fluents' requirements. All unique object names of cause and effect events in a TAC rule are encoded as variable names and types. While the objects describe the entities involved in the occurrence of an event, defining

variables with types will also inform the particular entities required to perform the domain action. All variables that are used in the domain model have types, hence the need for typing requirement. Using the example, a typed variable would be '?DisplayName - DisplayName'. Furthermore, every TAC rule has three numeric values: counter, temporal-association accuracy and causal rank. The representation of these numeric constraints in the PDDL domain model needs fluent requirement.

4.3.2.2 Predicates

The predicates represent the set of unique events found in the TAC rules. A predicate is a combination of event type and its corresponding parameter list. It is formatted as '(e_<EventType> TYPED-PARAMETER-LIST)'. Using the example TAC rule, '(e_4720 ?DisplayName - DisplayName ?HomeDirectory - HomeDirectory $\cdots$?UserWorkstations - UserWorkstations)' is a predicate to model event type 4720. Notice that the variables and types have same names, except the variables starts with question mark symbol ('?').

4.3.2.3 Functions

A function or numeric fluent is way of storing numeric values in the domains actions, initial state and goal state, and the values are accessible during the execution of planning algorithm. A function named `accumulative-weight` is used in the domain model to hold the amount of confidence or importance of every modelled rule. This function is increased upon the selection of underlying action by a value that is the sum of counter, temporal-association accuracy and causal rank values of the rule. Using the example, the TAC rule ($4720 \rightarrow 4673$) has a sum of $3 + 95 + 3 = 101$.

4.3.2.4 Actions

The actions are responsible for changing the current state of the world. In our research, the cause and effect relationship between events in the TAC rules is represented as a action, and there are same number of domain actions as there are number of rules in the database. An action has four components: name, parameters, precondition and effect. The name of an action is formatted as {e_<Cause event>-to-e_<Effect event>}, e.g.

e_4720-to-e_4673. The combined list of unique object names from both cause and effect of a TAC rule constitutes as a parameter list. Each domain action encodes all objects (in the form of one or more parameters) that are required to make any configuration change in the underlying system.

The precondition and effect are always written as *and*/*or* conjunctions. The precondition is composed of a single predicate that models the cause event, whereas the effect part represents the effect event of a TAC rule, as well as the accumulative-weight function that assigns a cost/confidence value to the action. A negative predicate is also placed in the effect that prevents the consecutive selection of the same actions, involving the same objects. It is false (void or inapplicable) before the action selection and means that the given arguments have not been processed yet, and consequently marked true to depict that the same arguments cannot be executed again. Note that the negative predicate would not limit the repetition of actions in a plan. It would just prevent successive redundancy that can occur due to absence of goal state and use of maximisation metric in a problem instance (explained in Section 4.3.3.3). So the planning algorithm would not allow the duplication of the same set of actions to forge a maximum accumulative-weight value, and only generate a precise plan of actions.

4.3.2.5 Domain Model Example

Continuing the example, a complete domain action model of the TAC rule is provided in Figure 4.2. The event types 4720 and 4673 have some objects in common, e.g. 'SubjectDomainName' and 'SubjectUserName'. Such duplicate objects will be removed for the parameter list.

4.3.3 Problem Instances

Problem instances are automatically constructed using a given domain action model and the event log entries of a vulnerable machine. A problem instance contains the following components: objects, initial state, goal state and an optimisation metric. The reason behind using the domain model is to ensures that the object types and initial state of a problem instance are restricted to the types and predicates of the domain model, thus avoiding any execution errors during the planning.

```
(define (domain EventRelations--1)
   (:requirements :typing :fluents)
   (:types DisplayName HomeDirectory HomePath ObjectServer PasswordLastSet
      ··· TargetDomainName TargetSid TargetUserName UserWorkstations)
   (:predicates
      (e_4720 ?DisplayName - DisplayName
         ··· ?UserWorkstations - UserWorkstations)
      (e_4673 ?ObjectServer - ObjectServer
         ··· ?SubjectUserSid - SubjectUserSid)
   )
   (:functions
      (accumulative-weight)
   )
   (:action e_4720-to-e_4673
      :parameters (?DisplayName - DisplayName ···
         ··· ?UserWorkstations - UserWorkstations)
      :precondition (and (e_4720 ?DisplayName - DisplayName ···
         ··· ?UserWorkstations - UserWorkstations))
      :effect (and (increase (accumulative-weight) 101)
         (not (e_4720 ?DisplayName - DisplayName ···
            ··· ?UserWorkstations - UserWorkstations))
         (e_4673 ?ObjectServer - ObjectServer ···
            ··· ?SubjectUserSid - SubjectUserSid))
   )
)
```

FIGURE 4.2: An example of PDDL action, which is created from the temporal-association-causal relationship of two events. It also shows the relevant objects/parameters and an accumulative-weight value.

4.3.3.1 Problem Name And Objects

The problem instance is formatted as {Problem–<Domain name>}, e.g. Problem-EventRelations–1. Using object or property name-value pairs of the event entries, values will be used as objects, whereas the respective names will be assigned as types. Defining types improves the quality of domain model as it categorises the objects of problem instance. The planning algorithm uses these objects as constant arguments to the domain actions. This is quite beneficial for the non-experts as these objects will be involved in carrying out the security-related actions.

4.3.3.2 Initial State

The initial state models the current state of a vulnerable machine by using its security event logs. It is a set of predicates, whose variable parameters are initialised/replaced

with the corresponding objects or constants. Such predicates are known as *grounded* predicates. Each predicate represents a unique event type of a vulnerable machine. Only those predicates are included in the initial state, which also occur in the domain action model. In other words, all predicates of the initial state are also found in the domain model. The order of predicates (or events) is not important as they are considered as a collection in the initial state.

4.3.3.3 Goal State And Optimisation Metric

Defining a goal state in this particular case is a challenging task. First of all, we cannot determine or even predict beforehand what actions need to be scheduled. Secondly, there is no way of finding the set of one or all missing configurations in the target machine. Several scenarios can be speculated at this stage. Let us consider a scenario where a new predicate, say `Goal-reached`, is introduced. This predicate becomes part of every action's effect in the domain model as well as a goal state. Doing so will not work because as soon as the planner finds a single action with respect to the initial state, it would have reached goal and the planning process will be terminated. Again, consider a similar scenario where each domain action is assigned its own unique predicate, for example `Goal-reached-1`, `Goal-reached-2`, $\cdots$, `Goal-reached-N`. All these predicates form a goal state. This is not an acceptable solution as the planner would never reach the goal unless every single action is found to be included in the plan solution.

Considering the aforementioned discussion, the goal state is kept empty. Also, a maximisation plan metric is applied in the problem instance to obtain the complete set of actions or those with the highest accumulative-weight value in the specified planning time. As the PDDL does not allow more than one metrics in a single problem instance, we have used the sum of counter, temporal-association accuracy and causal rank as an accumulative-weight. A combination of empty goal state and maximisation metric enables the planning algorithm to explore all possibilities with respect to initial state and find a quality solution. The planning algorithm will seek for the maximum total accumulative-weight value of a plan, which is the sum of accumulative-weight of each chosen action. In other words, the planning algorithm will choose the maximum number of actions that have relatively higher accumulative values, hence finding an accurate and complete plan solution for a given problem.

```
(define (problem Problem-EventRelations--1)
   (:domain EventRelations--1)
   (:objects
      Test1 - DisplayName
      NULL - HomeDirectory
      NULL - HomePath
      DS - ObjectServer
      0_1794 - PasswordLastSet
      SeTcbPrivilege - PrivilegeList
      0x238 - ProcessId
      C:\Windows\System32\lsass.exe - ProcessName
      NULL - ProfilePath
      test1@local - SamAccountName
      NULL - ScriptPath
      WORKGROUP - SubjectDomainName
      0_0x3e7 0_0x105f5 - SubjectLogonId
      IE8SAAD$ IEUser1 - SubjectUserName
      S_1_5_18 S_1_5_19 - SubjectUserSid
      IE8Win7 - TargetDomainName
      S_1_5_32 - TargetSid
      Administrators - TargetUserName
      NULL - UserWorkstations
   )
   (:init
      (= (accumulative-weight) 0)
      (e_4720 Test1, NULL, NULL, 0_1794, NULL, test1@local, NULL,
         WORKGROUP, 0_0x105f5, IE8SAAD$, S_1_5_18, IE8Win7, S_1_5_32,
         Administrators, NULL)
      ⋮
   (:goal (and ) )
   (:metric maximize (accumulative-weight))
)
```

FIGURE 4.3: An example of problem instance generated automatically from a machine that has poor security configurations

4.3.3.4 Problem Instance Example

A problem instance from a live system is presented in Figure 4.3. It was extracted from a machine having poor security configurations. Every object is assigned to a type, 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 initial state is comprised of the grounded predicates, and the goal is not explicitly specified. The accumulative-weight value is initialised with zero and will be incremented (to maximum extent) according to the execution of domain actions.

4.3.4 Automated Plan Generation

The plans (sequence of actions) are produced by a planning algorithm or planner, which is capable of processing PDDL-based domain action model and problem instance files [178]. The purpose of a plan is to show the progression of actions to achieve the goal within specified constraints. In our context, a plan presents the series of steps for a particular machine that shows the security assessment process and proposes a solution. The planning algorithms can be categorised into two main types [179]: (1) guarantee to discover the optimal solution, provided they are given enough time and resources, and (2) generate the best possible plans within a predefined time with no assurance of optimality. The domain model used in this research involves large number of actions alongside numeric computations, and the problem instance might contain large number of objects as well. Furthermore, there is a limited amount of time and computing resources available for the planner. Hence, we have chosen a planning algorithm from the second category.

The remaining section reviews the problem solving techniques to obtain a quality plan of actions and also provides the concepts related to the general working of existing planning algorithms. After that, it explains the process of generating plan solutions with an example.

4.3.4.1 Planners

The first problem-solving planner, called Nets of Actions Hierarchies (NOAH), was developed in 1975 [180] that presented a practical implementation of conjunctive goals and treated them as independent and additive. After that, a system known as Graphplan [181] was developed in 1997 that converted STRIPS-style specification into a planning graph, and then applied a graph analysis technique (satisfiability planning) to produce partial-order plan solution. These solutions represent a sub-graph of the planning graph, contains all the facts from the initial and goal states, and might have several parallel predicates occurring at the same time step. The main focus of Graphplan was to increase the planning productivity and efficiency by generating the shortest possible plan. Another similar system was developed in 1997 that integrated a stochastic/random search algorithm into the satisfiability planning and improved the scalability issues [182].

One of the most successful planner was introduced in 2001, called Fast-Forward (FF) [183]. It proposed a new forward-chaining approach that unified a variant of hill-climbing approach with the Graphplan. The hill-climbing is used to perform a local and systematic search, whereas the Graphplan acts as a heuristic estimator function that reduces the search space. The FF planner also includes goal prioritisation mechanism that helps in saving time on achieving goals that should to be considered in later stages. The representation of numeric constraints is essential for modelling real-world problems. For this purpose, the FF was extended to Metric-FF [184] planner in 2003 that allowed the use of numeric constraints to enable arithmetic operations and the user-specified optimisation metric, i.e. minimisation and maximisation of action costs for improving plan solution quality.

Another planner, called Local search for Planning Graphs (LPG) [185], was introduced in 2003 that combined stochastic and local search algorithms to support the features of PDDL 2.1. The LPG is a domain-independent planner that employs an informed search technique (described in previous Section 2.4.2.2). It is a winner of both third and fourth IPCs. It is a complete planner but does not guarantee an optimal output. It is also the first incremental planner that generates more than one valid plans, each of which has improved the plan's quality over the previous one. The plan quality is flexible and defined through multiple criterion, such as number of plans to generate, time restrictions and computing resource allocation. The LPG planner is also capable of utilising best-first search to determine the search and execution costs of achieving a precondition. These costs are calculated by using heuristic estimator functions [186]. Another important feature of the LPG is a better performance on large-scale planning problems [187], which is achieved by creating a balance between finding quick and best solutions.

Exploring a complete list of planners is beyond the scope of this thesis. However, the above discussion shows that various planners are available to satisfy our requirements, i.e. heuristics-based search planning, scalable to manage large datasets, and process PDDL-based numeric fluents and optimisation metrics. For this research, we have adopted the LGP planner to extract the ordered set of actions from our domain model and problem instances that can improve the security of vulnerable machines. The order of plan actions is important to perform the configuration tasks. The main reason of using

```
[0]:  (E_4720-TO-E_4673 Test1, NULL, NULL, DS, O_1794, SeTcbPrivilege,
       0x238, C:\Windows\System32\lsass.exe, NULL, test1@local, NULL,
       WORKGROUP, O_0x3e7 O_0x105f5, IE8SAAD$ IEUser1, S_1_5_18 S_1_5_19,
       IE8Win7, S_1_5_32, Administrators, NULL)
```

FIGURE 4.4: A simple plan generated by by the LPG planner

LPG is to exploit its incremental planning capability for identifying plans of increasing quality within a user-specified time.

It should be noticed here that we can void the stochastic nature of the LPG planner by adding a new predicate, say `Check-all-Objects`, to the precondition of every domain action, and initialised in the problem instance. As this predicate will be true from the beginning of planning, it will enable the planner to consider all objects and actions before reaching a decision, regardless of the initial state. However, for a large-scale domain model, this will require a significant amount of time and computing resources that might not be always available. As this research is aimed at building an efficient solution, we did not apply this technique.

4.3.4.2 Example

Using the simple domain action presented in Figure 4.2 and the problem instance provided in Figure 4.3, the LPG algorithm will generate a plan solution as shown in Figure 4.4. Although the plan is quite simple, it describes the plan generation process. The objects correspond to the parameter list of the domain action and describe the information that is required to perform the proposed operation. The plans shows that a user account `IE8SAAD$` was created in a network by another account `Administrators`. The `Administrators` account performed the process with 'SeTcbPrivilege' permission, which means it had system-wide authorisation to access any resource.

There is a possibility of multiple actions having same a precondition but different effects. In that case, the action with highest accumulative-weight value will be included in the plan and the rest of them will be discarded. Still, if the accumulative-weight is identical as well in such actions, the LPG planner will employ the heuristic function to choose the action that, in later stages of planning, will maximise the overall accumulative-weight value.

It should also be noticed here that the plan solution includes only 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 concerns or some missing configurations, while the remaining part manifests the mitigation actions. As the routine/meaningless events are filtered in early stages of creating TAC rules 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 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 provide useful insight to some extent for the non-experts. 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.

4.4 Chapter Summary

This chapter describes the representation and utilisation of knowledge acquired as the TAC rules. This chapter begins by describing the conceptual model of classical planning along with general assumptions. After that, it explains the process of knowledge representation and domain modelling. The chapter also presents an overview of common domain representation languages.

The representation of TAC rules is performed in two fully automated steps: (1) encoding the rules into a PDDL domain action model and (2) converting event log entries of a given vulnerable machine into a PDDL problem instance. After that, the domain model and problem instance are used as input for the LPG planner to search a plan solution in reasonable time. The plan is complete and sub-optimal, and it proposes the sequences of actions that can provide security assessment and configuration steps relevant to the vulnerable machine.

As mentioned before, the AP techniques have been successfully used to conduct penetration testing on systems and networks. The application of AP is also integral in our research as it provides several benefits, which are provided in the following list:

1. AP algorithms provide a fully automated and systematised deliberation mechanisms to replicate the human expert actions;

2. The parameters and accumulative-weight values of domain actions enabled the modelling of additional information specified in the TAC rule;

3. Objects in the actions of a plan belong to the underlying machine, hence providing usable and relevant results for non-experts;

4. Ability to generate an optimal plan based on a metric, ensuring that actions with the maximum accumulative-weight values are chosen to output the best results against a given situation; and

5. The knowledge is presented in a standardised format and can be utilised by any PDDL supporting planner, and thus distributing a benchmark domain model for the AP community.

Chapter 5

Implementation and Evaluation

This chapter provides the details of proof-of-concept implementation along with the evaluation process of the solution proposed in this research. The implementation includes the explanation of complete software development process, which is supported by figures and and other visual aids for evidence and better understanding of the reader. The software application is based on the theoretical foundation of the proposed solution, which is provided in the previous chapters. After building the application, it is evaluated against the security expert knowledge to determine the quality, accuracy and validity of the underlying concepts. This is based on the empirical analysis of evaluation data and further includes performance and scalability comparison with manual security assessment and configuration procedures. This chapter also discusses the limitations and bottlenecks of the developed application in terms of live operation, environment and usage.

5.1 Implementation

This section presents the specific details of software requirements, design and implementation of the proposed solution. Various aspects were considered during the design and development stages of the solution, such as ease of usability for all users having different technical knowledge, robust and suitable programming languages, clear and precise format of input/output, error and exception handling, and higher flexibility. It is important to use a structured approach to build the proposed solution as any logical

error/bug in the application will lead to the incorrect output, and negatively impact the quantity and quality of final results.

5.1.1 Software Requirements

The proposed solution is implemented with the help of two separate, prototype software applications. The requirements of both software applications are described in Chapter 3 and Chapter 4. To get a better understanding of the overall scheme, Figure 5.1 shows all components of both applications: knowledge acquisition ($1A$) and representation ($1B$), and knowledge utilisation (2). This figure summarises the steps taken from processing the event log entries to producing a security action plan. Both applications should be fully automated and fulfil all functional requirements. The synopsis of each application is described in the following:

1. An application to perform the *knowledge acquisition and representation* process, which consists of the following two modules:
   (a) For the first module, read the event log entries of a machine that was configured by security experts, remove the routine events and acquire the expert security knowledge by learning user-activity patterns in the form of TAC rules;
   (b) Convert the TAC rules into a PDDL domain action model file, such that it represents the acquired knowledge from a particular event log dataset. The users should be able to enable or disable this feature;
   (c) Store every TAC rule in a MySQL database. Each rule should have 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; and
   (d) For the second module, collect all TAC rules from the database that were acquired from one or more distinct event log datasets and convert them into a single PDDL domain action model file to represent and utilise the acquired knowledge.
2. The second application will perform the *knowledge utilisation* process, which also consists of two modules:

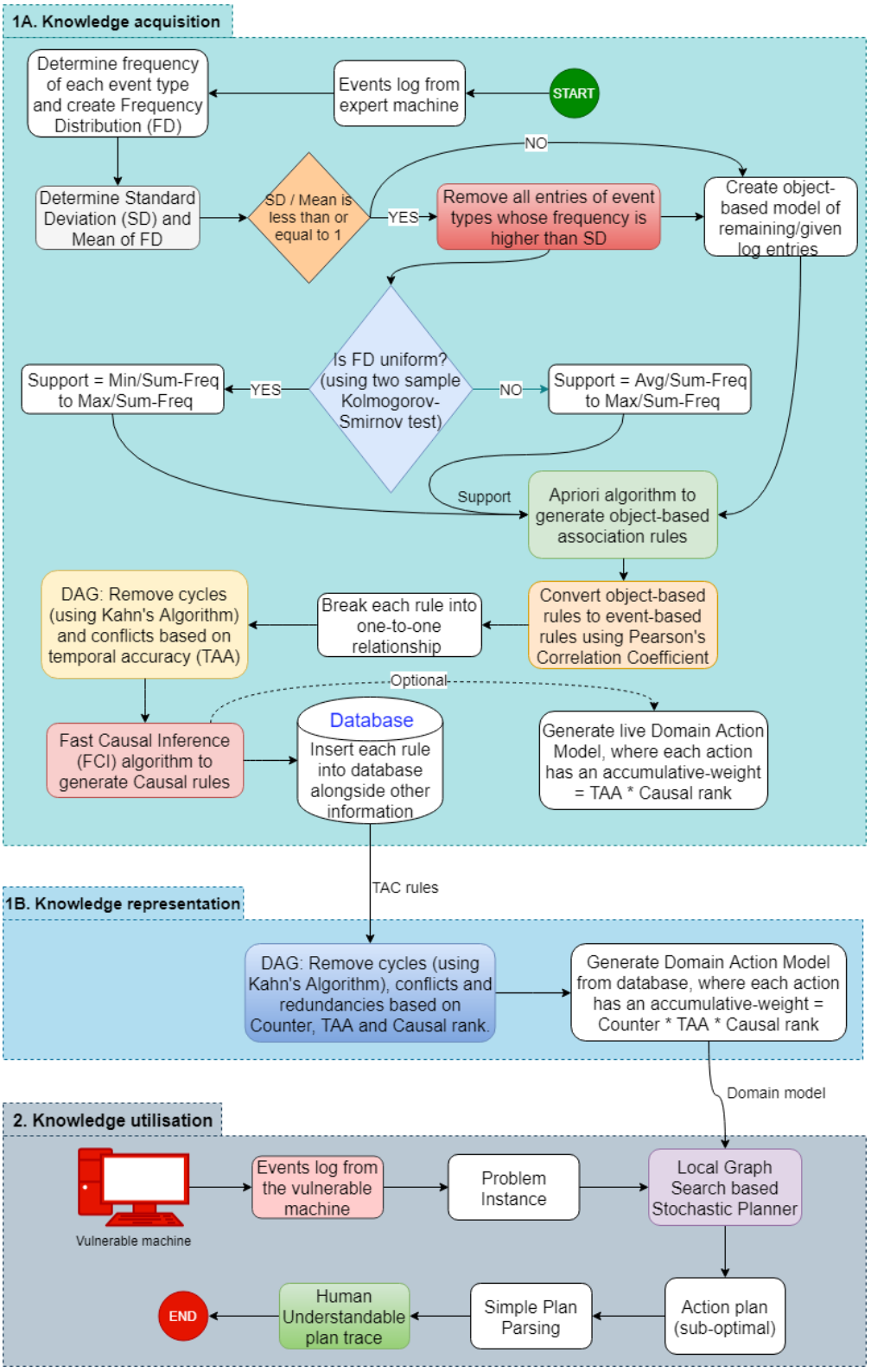

FIGURE 5.1: Overview of the proposed solution represented in two phases. The first phase is further divided into two segments; first, extract temporal-association-causal (TAC) rules from the given event log dataset and second, represent the TAC rules in a PDDL domain action model file. In second phase, a plan solution is generated for the vulnerable machine to improve its security.

(a) For the first module, read the event log entries of a previously unseen, vulnerable machine along with the existing domain model to generate a PDDL problem instance;

(b) Modify the default parameters of existing LPG planner software tool (details are in Section 5.1.2.2) to accommodate the processing of domain models and problems instances with a large number of objects, and use it to generate a sequence of actions that can be used to perform security assessment and configuration on the vulnerable machine; and

(c) For the second module, create a simple parsing tool that essentially converts the output plan solution of LPG planner in a clear, descriptive format for presenting the discovered security risks and proposed mitigation actions to non-expert users.

Flexibility is one of the significant requirements for building the application. This is because the working of some features needs exploration before finding the best performing implementation methodology. This will also allow the users to configure the software application in a desired manner. For example, the proposed solution calculates the support range automatically for Apriori algorithm (Section 3.2.2); however, users should also be allowed to manually input the minimum and maximum support values in case of incorrect automated values. Another example is the elimination of temporal-association rules based on a 50% threshold of temporal-association accuracy value (Section 3.3.1). This threshold should be changeable by the user to achieve a desired balance between the quantity and quality of rules.

Another requirement is the design of both applications that should cater the need of all the potential users. For the knowledge acquisition and representation application, the main focus should be on the expert-level users, who want to acquire domain action models from configured machines. For the knowledge utilisation application, the main focus should be on the non-expert users, who want to analyse and understand the security status of vulnerable machines. Furthermore, in order to provide a better experience to all kinds of users, both applications should display and explain the step-by-step output of each module in a well-known and clear format, e.g. within console and HTML. This is not an essential requirement for the developed solution to operate; however, it

can benefit both expert and non-expert users in terms of debugging and visualising the results of overall process as shown in Appendix C.

The proposed solution should also aim to reuse the open-source code of existing tools and libraries during the development of both applications (where possible). Such code is generally reliable, tested, verified, crowd-sourced and will also minimise the development effort [188]. For example, the Two-Sample Kolmogorov–Smirnov normality test, performed in Section 3.2.2.1, has already been implemented in Accord.NET[1] library. Similarly, the source code of graph analysis and several causal inference algorithms used in Section 3.4 is available in a Pcalg[2] library of R language. Furthermore, the source code of LPG planner[3] used in Section 4.3.4.2 is also available and can be easily modified to fit our needs. The format of input and output of different components in both applications are designed to be compatible with external tools.

5.1.2 Development Of Applications

Both applications have been developed using standard software engineering techniques. The source code is written in the C#, Python and R programming languages. The C# is used for graphical user interface, reading and processing event log entries, and generating domain models and problem instances. The Python is used to implement Apriori algorithm for association rule mining. The R language is used for causal analysis by implementing Fast Causal Inference algorithm. Both applications are configured as desktop-based and contain several modules, where every module is responsible for a specific operation. Both applications are low in coupling and high in cohesion. The coupling refers to the degree of different modules depending on each other, whereas the cohesion refers to the degree of elements within a module belonging together [189]. Therefore, all modules are independent of each other, and at the same time all related code is joined as close as possible. This approach also ensures high 'Separation of Concerns', where every module can easily be modified without affecting the workings of other modules in the application.

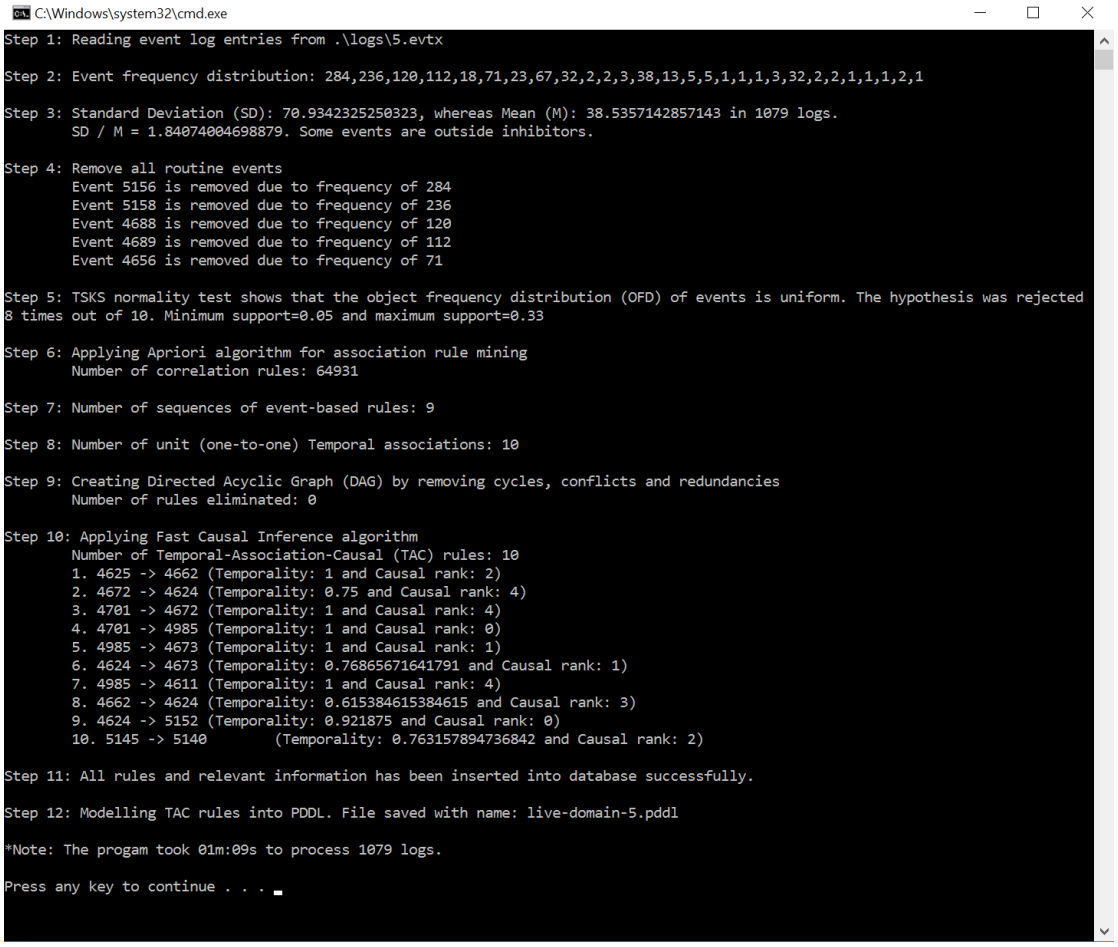

FIGURE 5.2: Screenshot of console application demonstrating the generation of (live) PDDL domain action model using event log entries. This domain model only represents the rules extracted from a single event log dataset.

5.1.2.1 Knowledge Acquisition And Representation

The knowledge acquisition and representation tasks are performed through a command-line application. It has two parts that perform two distinct operations. First, generate temporal-association-causal (TAC) rules directly from the event log entries and store them in a database along with encoding the rules into a PDDL domain action model. Second, generate a domain action model from the database by consolidating all stored rules.

Figure 5.2 demonstrates the execution of first part of this application using an event log file, named '5.evtx', and producing a domain action model file, named 'live-domain-5.pddl'. It mainly consists of twelve steps. The input event log dataset contains 1,079

[1]http://accord-framework.net/docs/html/T_Accord_Statistics_Testing_KolmogorovSmirnovTest.htm
[2]https://cran.r-project.org/web/packages/pcalg/index.html
[3]http://zeus.ing.unibs.it/lpg/

entries. The application extracted all unique event types, which are 28 in number, and created an event frequency distribution (FD). The Standard Deviation (SD) and Mean (M) of FD was found to be 70.93 and 38.54, respectively. As the coefficient of variation ($\frac{SD}{M}$) is greater than 1, 5 event types were removed as their frequencies were higher than the SD. After that, the application created the object frequency distribution (OFD) of the remaining entries and conducted a Two-Sample Kolmogorov–Smirnov normality test. The OFD was determined as normal distribution and the support range (SR) was calculated as $5\% - 33\%$. Using the SR and 100% confidence value, the Apriori algorithm generated 64,931 object-based association rules. These rules produced 9 chains of events, which resulted in 10 temporal-association rules using a 50% threshold of temporal-association-accuracy value. The entire set of these rules formed a Directed Acyclic Graph (DAG) as no cycles, conflicts and redundancies were found. After that, a causal rank was calculated and assigned to each rule in the DAG using Fast Causal Inference algorithm. This produced the final set of TAC rules, which was stored into the MySQL database and represented as a PDDL domain action model as well. Every step of this application is performed in a fully automated manner, and are also capable of error management and exception handling. It is worth mentioning here that the processing of this application is based on batches. It can process a batch of event logs, specified by a directory path, and generate individual domain action models against each dataset.

Every time an event log dataset is processed, the resulting TAC rules alongside other relevant information are stored in the database. Another Figure 5.3 presents the second part of the same application. It shows the process of generating a PDDL domain action model file, named 'database-to-domain.pddl', directly from the database that contains 347 TAC rules. After creating a DAG of the rules, only 134 of them remained and were encoded into a domain action model. Rest of the rules were eliminated by the application due to cycles, conflicts and redundancies.

5.1.2.2 Knowledge Utilisation

The knowledge utilisation phase is based on three steps: first, generate the problem instance; second, produce a plan solution; and third, convert the plan into a user-friendly

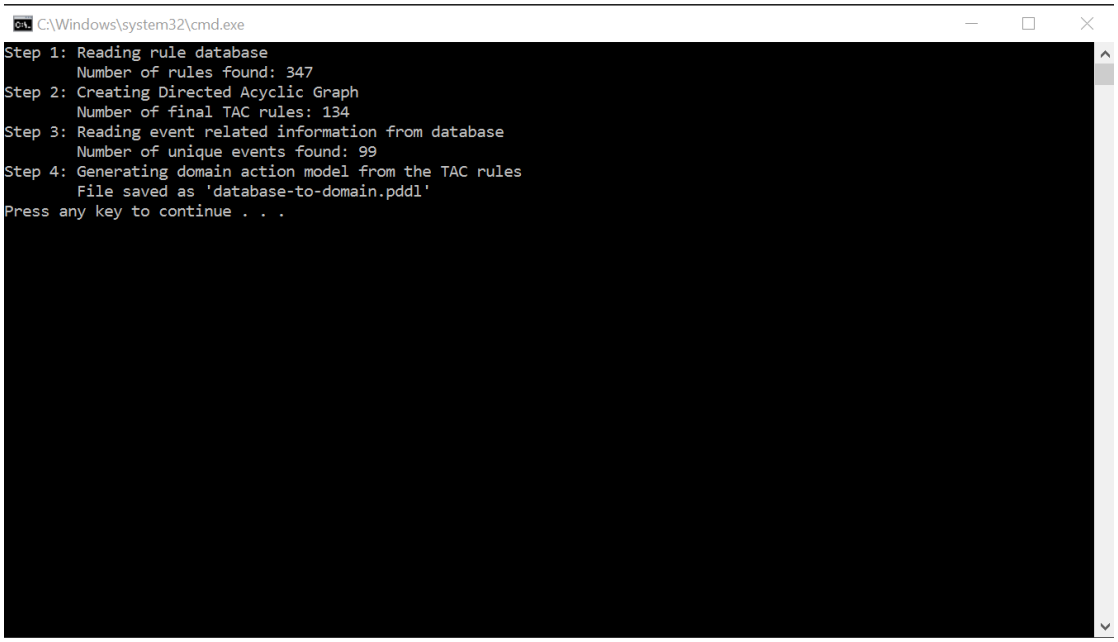

FIGURE 5.3: Screenshot of the same application demonstrating the generation of PDDL domain action model from the existing rules stored in database. This domain model represents the set of all those rules, which were previously acquired from one or more event log datasets.

format. The knowledge utilisation mechanism is implemented through a graphical user-interface application that guides the user to appropriately perform all steps in an ordered manner.

The first part of the application takes event log entries along with the domain action model that contains the security expert knowledge, and produces a problem instance as shown in Figure 5.4. It is possible that the objects extracted from the event log entries and used in the problem instance do not align with the input format of planner, and hence will require alteration. For example, objects starting with a number or fully numeric objects are not acceptable, so we added a 'O_<numeric object>' prefix to denote them as alpha-numeric objects. Similarly, some objects might contain special characters, such as round brackets, colons, commas and so on, which are not permitted. Such characters raise syntax errors during the planning process and will be completely removed if found in any object.

The second part of the same application is to use the domain action model and problem instance as input to LPG planner and produce a plan solution. The LPG produces multiple plans as it is an incremental planner, where each increment or iteration has increased quality. Therefore, only the last plan is displayed as an output as it would be

FIGURE 5.4: Screenshot of application that takes the domain action model and event log entries of a vulnerable machine to generate a problem instance, representing the current security state of the machine.

of the highest quality. As the source code of LPG is available, we increased the value of constants *MAX_RELEVANT_FACTS* to 40,000 and *MAX_TYPE_INTERSECTIONS* to 10,000 and recompiled the code to accommodate all problem instances, especially the ones with higher number of objects. The developed application limits the maximum amount of plans to 15 and time to 30 minutes. The reason behind enforcing 15 plans limit is to find as many plans as possible; however, this limit is flexible and can be any number from 2 and onward. The 30 minute execution time of the planner is programmed

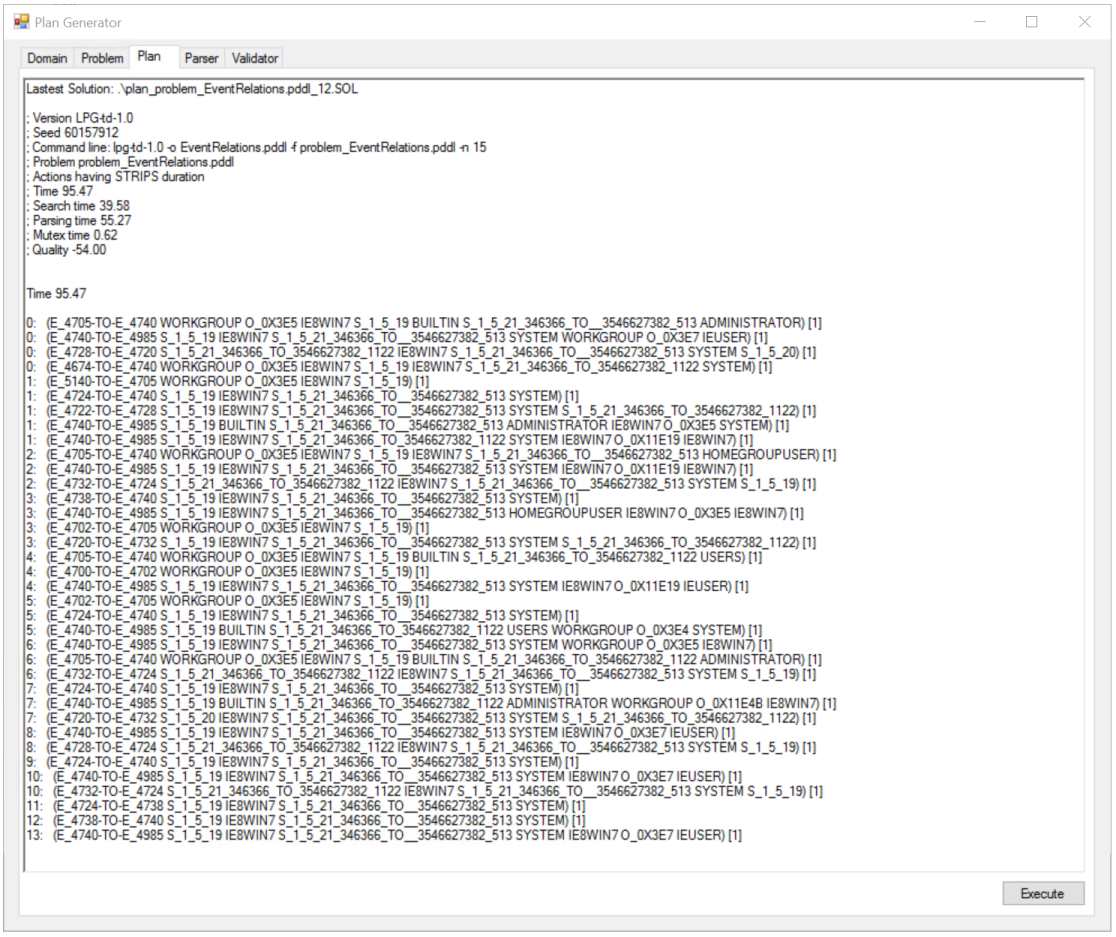

FIGURE 5.5: Screenshot of the same application that takes the domain model and newly generated problem instance to produce an action plan for expert security assessment and configuration actions.

by-default and can be changed as well. Due to these limits, the execution of planner will automatically terminate if it either finds 15 plans or exceeds 30 minutes of planning process. The user can also manually signal the planner to exit. An example of plan is shown in Figure 5.5, which is the 12^{th} plan solution. This plan was found in 95.47 seconds, which means that the planner executed for 30 minutes and could not find more than 12 plans. It should be noticed here that there is a repetition of actions in the plan. This is due to the planner providing a collective solution for multiple objects of the same types.

The final part of the same application is to convert the planner output into a more usable and explanatory format, as shown in Figure 5.6. This is achieved by implementing a simple plan parser that splits every action name into separate cause and effect events, along with adding their event descriptions. The parser processes the entire plan to describe the security issue and suggested solution. The order of parsed plan is kept same

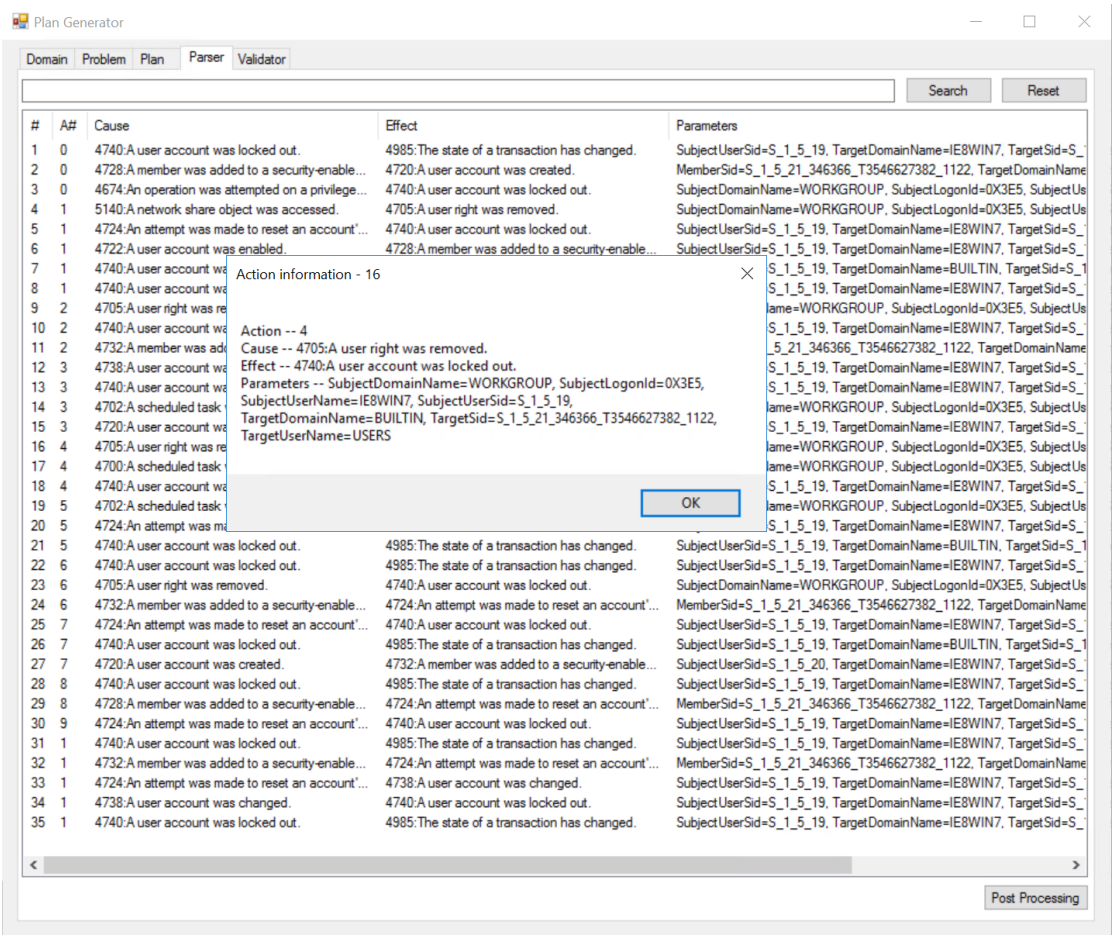

FIGURE 5.6: Screenshot of the same application that contains a simple plan parser to display the LPG planner output in an easy-to-understand format.

as the original plan. Also, every object-value in the action parameter list is assigned a corresponding object-name to make it more descriptive. The object-names are extracted from the given event log entries. The plan parser also includes a searching mechanism that uses a simple string matching as well as regular expressions to find keywords in event types, descriptions and object-value pairs.

5.2 Evaluation

This section presents an evaluation framework to explore the accuracy and performance of the implemented solution. This involves empirical analysis of the automated learning behaviour of the developed solution using test data. The implementation is a prototype

of this research and has been developed with the intention to assess the proposed technique from multiple aspects. The aims of the evaluation framework are stated in the following:

1. Test the ability of the developed solution to automatically discover the patterns of event relationships, which accurately depict the expert user actions for security improvements; and
2. Determine the efficiency and performance of using automated over manual plan solution in identifying security issues of the underlying machine and proposing mitigation actions.

5.2.1 Challenges

According to a recent study [190], there is a critical deficiency of security event log datasets that are operational and publicly available, especially from Microsoft Windows-based operating systems. This is due to the security and privacy related concerns of sharing data. The available datasets have limited access and other constraints regarding their usage. Most of the datasets are not suitable for research purposes as they do not provide a complete understanding of the context/environment and information about the user activities that triggered the events. This can lead researchers to forge assumptions about the fidelity and utility of event log datasets. So, one of the main challenge in evaluating the developed solution is to find or rather produce benchmark datasets that contain events depicting mitigation action(s) for predefined security vulnerabilities. Generating such datatsets would need a thorough understanding of event logging mechanism and security auditing and policies, along with the expert security knowledge. Without understanding the knowledge inside the datasets, it would be quite challenging to evaluate the extracted sequences of event relationships.

The PDDL domain action model, which represents Temporal-Association-Causal (TAC) rules acquired from the event log datasets, acts as a knowledge base for generating plan solutions against specified problems. Therefore, the correctness of the domain knowledge and modelling process is a critical factor to determine the overall quality of problem solving process. According to multiple research studies [191, 192], the validation and verification (V&V) of planning systems are one of the biggest challenges. In our context,

verification is the process of finding whether the actions of a plan are correct with respect to the expert security knowledge, whilst, validation is the process of finding whether the actions of a plan are correct with respect to the domain model.

As mentioned in Section 4.2, the V&V process is performed based on the ability of operating in real-world settings, completeness and accuracy measures. Some tools are available that can validate a plan solution against the respective domain model, such as VAL [193]; however, there is no standard and dedicated process, automated or otherwise, which can be used for verification. The knowledge acquisition and representation are not mathematical procedures, thus a domain model cannot be measured on any accuracy scale. To encourage automated planning researchers, the International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS) has devoted a section to present novel domain debugging, validation and verification mechanisms. It is concluded in recent ICKEPS that the verification process is still relied on human expertise, and there is a lack of cooperation and coordination between the experts and domain modelling process [194].

The developed solution is the first system of its kind in terms of integrating automated rule mining, domain modelling and knowledge utilisation mechanisms. The previously proposed solutions employ human experts to define the event relationships, create and encode rules, and then perform automated or semi-automated deliberation process. As the aim of our solution is to replicate human expert actions for security assessment and configuration of a machine, it would require a human expert to decide whether the proposed sequence of actions for the problem instance is correct in terms of security resolution. Therefore, accuracy analysis of the solution would require human assistance.

5.2.2 Methodology

The evaluation methodology adopted in this research is inspired from the testing of several automated knowledge acquisition techniques [195, 196]. An overview of the methodology is shown in Figure 5.7, and described in the following:

1. Obtain the event log datasets, where they are already known to contain events describing expert mitigation and configuration actions for specific vulnerabilities;

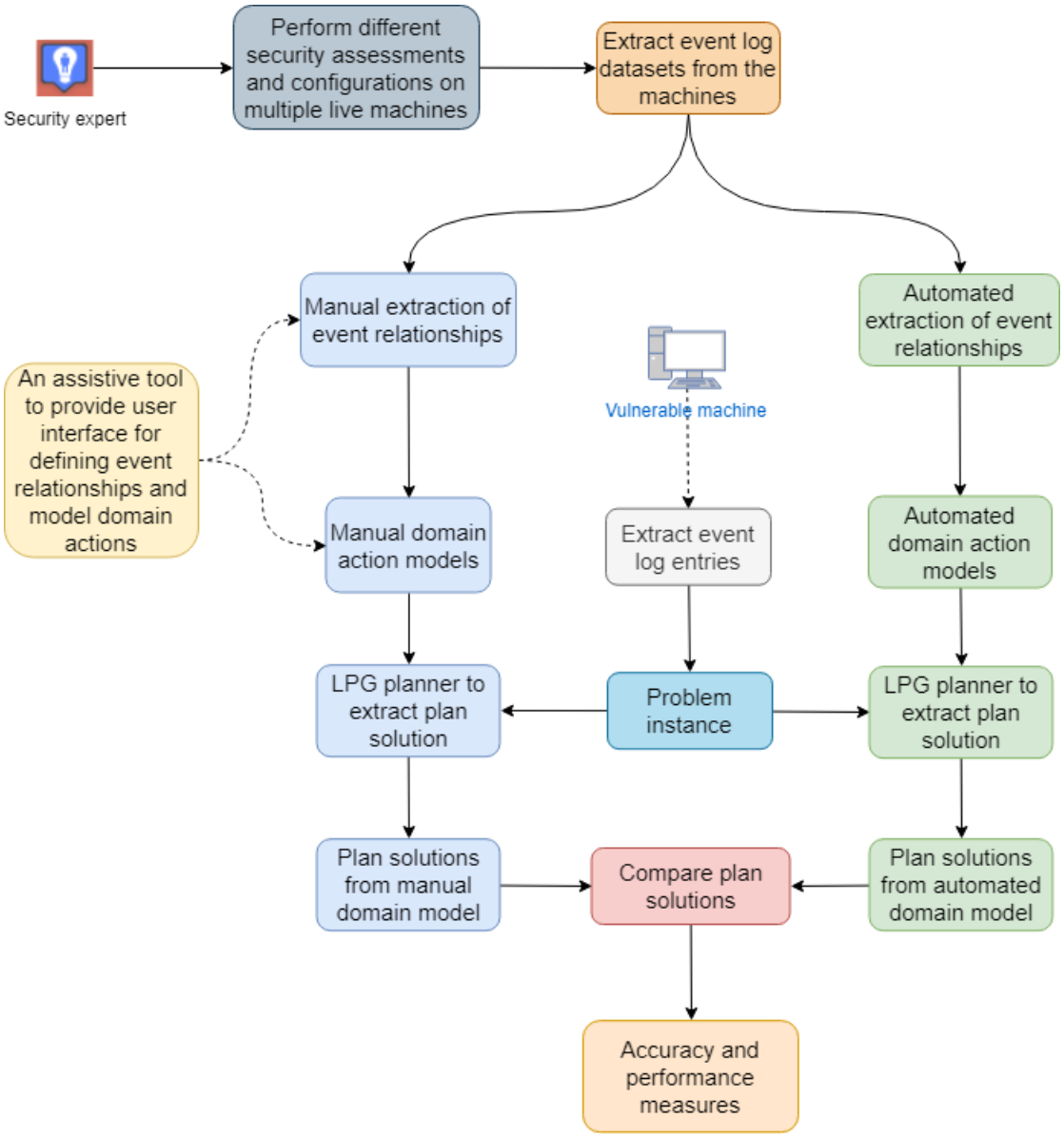

FIGURE 5.7: Overview of the evaluation methodology

2. Using the datasets, generate automated domain models from the developed solution and manual domain models with the help of human experts;

3. Extract a problem instance from a machine that is known to be vulnerable;

4. Find plan solutions from both automated and manual domain models against the same problem instance; and

5. Conduct a comparison between manual and automated plan solutions with respect to sequences of actions and time to determine accuracy and performance, respectively.

5.2.3 Process Of Evaluation

The developed solution has been evaluated using a set of security event logs acquired from multiple, live computing machines. Here the term 'live' is used to express that the machines are regularly used in the real-world operations and not solely for research purposes. The reason behind including multiple machines in the evaluation process is to verify that the developed solution is sufficiently adaptable and applicable under different circumstances. The accuracy of the developed solution is based on whether it was successfully able to extract knowledge from the configured machines, and then use the knowledge to propose human-like security actions for unseen vulnerable machines. This evaluation also determines the effectiveness and performance against a manual log analysis approach that requires significant security knowledge and human resources in terms of time and effort. It should be noticed here that the measured time displayed in the evaluation results is subjected to change. It depends on the hardware specification of a machine, and availability and amount of free computing resources. The remaining section provides a detailed explanation of the evaluation process.

5.2.3.1 Data Collection

To acquire data for evaluating the solution, 21 machines are chosen that are part of a university network and used on a daily basis by different kinds of users. All machines are configured by security administrators according to the university's policies. In addition, a tool called, Microsoft Security Compliance Toolkit[4], was used to confirm the implemented policies. This tool performs the comparison of actual and expected machine configuration based on the given security policies, hence verifying the ground truth. These machines have different operational roles/settings (e.g. student, staff, administrator etc.) and configurations. Each event log dataset has a distinct set of characteristics, such as amount of routine events, security-related activity, number of entries and so on. As the specifications of security policies are already known, their interpretation provided the ground truth and made it possible to determine the accuracy domain model generated by the proposed solution.

[4]https://www.microsoft.com/en-us/download/details.aspx?id=55319

5.2.3.2 Knowledge In The Events

As described in Section 1.2, the security event log of a Windows-based system contains a complete and organised information regarding the security-related activities of applications, system and users. Event logs are stored in local files and contain a repository of different activities, such as adding, deleting and updating the firewall rules, modifying the access permissions of a file or application, all attempts to log on a machine, change in system configurations, escalating or deescalating rights of user accounts, cryptographic operations, etc. An ordered collection of such activities creates a complete and holistic view of a security-related operation, thus providing beneficial knowledge for conducting security assessment and system configuration. A few examples of the patterns that exist in the 21 datasets are shown in Table 5.1. The table provides an evidence of how a sequence of events can describe a security-related operation that includes the identification of a security issue and its remedial action. Extracting such patterns in an automated manner and applying it to vulnerable machines will enable the users to perform expert security configuration, without having significant security knowledge.

5.2.3.3 Automated Plan Generation

To generate the plan solutions, the first step is to extract the security event logs of 21 Microsoft Windows-based machines, which are in the form of '.evtx' files. The second step is to process each file using the developed 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 evaluate the developed solution under adverse circumstances, and determine if the solution can recognise all security issues. 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, which is based on the ability of discovering correct set of vulnerabilities and suggesting mitigation actions for the machine.

Description	Events pattern
A user account was locked after several failed logon attempts. The account was deleted	1. 4625: An account failed to log on; 2. 4740: A user account was locked out; and 3. 4726: A user account was deleted.
An administrator probed the system for user accounts that have no password. Upon finding such an account, the user password was changed from empty to a certain string	1. 4797: An attempt was made to query the existence of a blank password for an account; 2. 4724: An attempt was made to reset an accounts password; and 3. 4624: An account was successfully logged on.
An unauthorised user account altered the security configuration settings of the machine. The user account was identified and suspended	1. 4673: A privileged service was called; 2. 4891: A configuration entry changed in Certificate Services; 3. 4739: Domain Policy was changed; and 4. 4725: A user account was disabled.
The remote security access was enabled on the machine, and several system objects were accessed by a user account. The remote access was removed along with the user rights	1. 1149: A successful RDP network connection was established; 2. 4624: An account was successfully logged on; 3. 4656: A handle to an object was requested; 4. 4663: An attempt was made to access an object; 5. 4705: A user right was removed; and 6. 4729: A member was removed from a security-enabled global group.
Critical services like Firewall, event logging and IPsec failed to start on the user logon. All services became operational	1. 4712: IPsec Services encountered a potentially serious failure; and 2. 4709: IPsec Services was started. *Or* 1. 5030: The Windows Firewall Service failed to start; and 2. 5024: The Windows Firewall Service has started successfully.

Table 5.1: Examples of security-related operations expressed in terms of order set of events. The events are displayed as <Event type>: <Description> format.

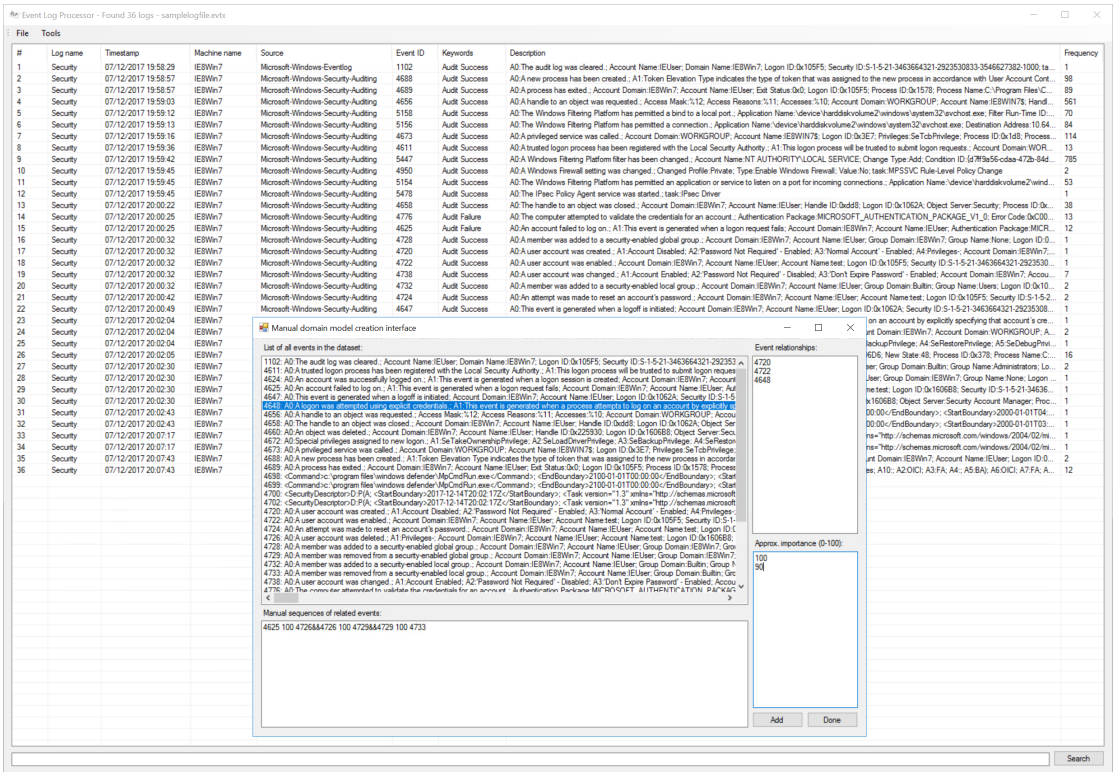

FIGURE 5.8: Process of manually extracting the knowledge from an event log dataset. The user can view all event entries along with their types, descriptions and objects. The event relationships and corresponding 'accumulative-weight' values are defined that are further encoded into a domain action model using PDDL.

5.2.3.4 Manual Plan Generation

Using the same set of security event logs, 21 domain action models are manually produced by three university security auditors. The auditors are experts in their domains and have a day-to-day job to monitor event logs and identify interesting patterns in terms of user activities. The reason behind using three experts to create a manual domain model is to perform a fair and impartial evaluation. A manual domain model, similar to automated domain model, also represents the set of actions that were taken by the security expert for either configuration or to resolve a particular issue in an underlying machine; however, the only difference is that the event patterns are extracted by the human experts. Processing a number of event log datasets, each with a large number of entries, can be a time-consuming, exhaustive and error-prone task. To make it less difficult, we developed another software application as shown in Figure 5.8. This simple application does not play any role in the proposed solution, and it is only developed for assisting the human experts in creating manual domain models. The feature set of this tool includes:

1. Load event log dataset from an underlying machine or a stored file;
2. View all event log entries including types, descriptions and all empty/non-empty object-value pairs;
3. Determine and display the frequency of all unique event types;
4. Search events based on a string or regular expressions matching. Both features are implemented in a single search bar;
5. Allow the user to define sequences of events along with the 'accumulative-weight' value for each event relationship; and
6. Producing a PDDL domain action model file based on the event relationships without manual encoding. The acquired PDDL domain is complete and without syntax errors, such that it can be processed by the LPG planner.

Using the manually constructed domain models and the problem instance, 21 plan solutions were produced. Each plan depicts the set of 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 implemented on the 21 machines) and expert opinions with respect to the security issues. Note that the large amount of time, effect and security knowledge is required to generate a manual set of security actions from large-scale event datasets. This further motivates the need for our proposed automated solution, where the complete process is accomplished without human intervention.

5.2.3.5 Accuracy

Accuracy is defined as a degree of similarity between a measured and a standard or accepted value. In our case, the accuracy of developed solution is determined by comparing the plan solutions of manually and autonomously acquired domain models. As mentioned before, each action in the plan solution involves action name and parameters. The action name represents the relationship between cause and effect events, whereas, the parameters represent the objects required to perform that particular operation. The comparison process determines the number of correct, incorrect, additional and missing actions of the automated plan, where:

1. An action of automated plan would be considered as 'correct' if the same relationship, i.e. have identical events and direction, is found in the manual plan as well;

2. An action of automated plan would be considered as 'incorrect' if the same events are discovered in the manual plan, but with an opposite relationship direction;

3. An action would be considered as 'missing' if it exists in the manual plan, but is absent from the automated plan; and

4. An action would be considered as 'additional' if it exists in the automated plan, but is not found in the manual plan. An important point here is that the additional rules, even if they are proved to be correct, reduces the accuracy of automated plan.

After finding these values, the second step is to apply the Equation 5.1 to calculate accuracy. Note that if the automated domain model is complete and accurate, the generated plans will represent similar steps that were taken by the human experts to perform a certain security assessment and/or configuration task.

$$Accuracy = \frac{correct}{correct + incorrect + missing + additional} \times 100 \tag{5.1}$$

It is possible that some of the extracted sequences of event relationships are obvious for the security experts; however, novice users might still be unfamiliar with them. The complexity and novelty of the knowledge presented by an automated plan is determined by the problem instance, i.e. amount and severity of vulnerabilities in a underlying test machine. If the machine has poor security, the automated plan would be large and complex due to suggesting multiple sequences of actions. Otherwise, the plan will involve relatively simpler and easy-to-configure actions. Further to that, the perception of complexity of the presented knowledge is subjective to the experience and background of the user. Henceforth, this empirical analysis only determines quantitative accuracy of the event patterns, rather than the qualitative accuracy. This accuracy measure does not examine how simple or difficult would it be for the user to implement the proposed security solution. It only determines whether the set of actions are capable enough to

discover the security issues and recommend one or more solutions for the vulnerable machine.

5.2.3.6 Performance

The performance of the developed solution is calculated by comparing the time required for producing automated and manual domain models from a given event log dataset. For automated domain modelling, the time refers to the overall execution time of the knowledge acquisition and representation application. The time is measured between reading the event log entries, discovering event relationships and the generation of the PDDL domain file. On the order hand, time required for the manual domain modelling process is the time taken by a human expert to manually probe the dataset and discover event relationships using the assistive application, as discussed in Section 5.2.3.4. The performance of the automated solution is demonstrated against the manual approach by comparing the number of entries in each event log dataset and corresponding time to create domain model from it. If the automated approach generates a domain model similar to the manual approach in less time, it will be deemed as more productive. Otherwise, if it takes more time, it will be considered as less effective.

5.2.4 Results

This section provides the evaluation results of the developed solution. The results are based on the empirical analysis of multiple datasets, and demonstrate the accuracy, scalability and performance of the proposed solution. The evaluation was conducted on a 32-bit Microsoft Windows-7 virtual machine with two cores of Intel i7 CPU at 3.50GHz and 4GB of RAM. All tools and applications described in previous sections are involved in the evaluation process.

Table 5.2 presents the results from each phase of processing for all 21 event log datasets. The table describes different kinds of information about each dataset, such as number of entries, computed minimum and maximum support values, number of rules, generated actions in the plan solution, etc. The table also includes the planner execution time of each domain model and problem instance, along with the percentage of accuracy using Equation 5.1. Notice that the time to produce any plan solution follows a consistent,

Event logs		*Rule Mining and Domain Modelling*						*Automated Planning*		
Event log dataset	**Number of events**	**Minimum Support**	**Maximum Support**	**Number of Object-based rules**	**Number of Event-based rules**	**Number of Temporal-Association rules**	**Number of Temporal-Association-Causal rules**	**Number of actions in plan**	**Planner execution time (seconds)**	**Plan accuracy**
1	5,027	0.07	0.59	516,608	14	84	32	37	123.03	87%
2	8,253	0.08	0.49	107,503	5	19	11	30	31	85%
3	521	0.25	0.49	147	6	11	9	5	10.55	83%
4	1,122	0.05	0.53	266,671	5	5	5	4	3.06	74%
5	1,079	0.05	0.33	64,931	9	17	10	9	6.19	88%
6	1,691	0.05	0.20	263,800	4	6	6	6	9.87	80%
7	1,714	0.13	0.46	978	9	58	21	32	72.64	87%
8	9,721	0.08	0.33	4,252	8	11	9	6	13.17	83%
9	9,770	0.05	0.32	4,027	9	21	12	5	5.38	73%
10	13,026	0.06	0.72	1,176	8	8	7	10	22.5	85%
11	10,948	0.05	0.62	1,349	5	33	11	9	8.69	85%
12	8,832	0.07	0.41	15,729	4	12	10	10	15.62	90%
13	3,403	0.05	0.37	97,574	10	7	6	9	8.27	78%
14	31,719	0.11	0.69	150,593	3	13	9	49	185.48	82%
15	1,072	0.07	0.26	14,681	7	15	11	6	6.62	86%
16	1,991	0.05	0.39	4,959	17	160	75	82	291.1	91%
17	826	0.05	0.40	280,482	4	8	5	6	8.27	81%
18	5,309	0.06	0.23	46,151	5	4	4	4	10.56	75%
19	69,854	0.05	0.37	66,600	10	16	15	72	263.55	92%
20	313,470	0.05	0.34	215	6	115	43	80	452.14	87%
21	32,688	0.15	0.43	237,155	4	6	6	7	6.69	80%

TABLE 5.2: Results of the empirical analysis of automated knowledge acquisition, representation and utilisation mechanism proposed in this thesis. The evaluation is performed on 21 different event log datasets obtained from live machines, which were configured based on expert security knowledge. The results are presented as output values from each stage of processing along with the amount of accuracy of each plan solution against a given problem.

proportional relationship with number of TAC rules, i.e. if there are more number of TAC rules, it will take more time and vice versa. The table presents the following specifications about the evaluation data and results:

1. Number of event log entries ranges from 521 to 313,470;
2. Minimum support value ranges from 0.05 to 0.25;
3. Maximum support value ranges from 0.20 to 0.72;
4. Number of object-based rules ranges from 147 to 516,608;
5. Number of event-based rules ranges from 3 to 17;
6. Number of Temporal-Association rules ranges from 4 to 160;
7. Number of Temporal-Association-Causal rules ranges from 4 to 75;
8. Number of actions in plan solutions ranges from 4 to 82;
9. Amount of time to produce plan solution by the planner ranges from 3.06 to 452.14 seconds; and
10. Amount of accuracy of plan solutions ranges from 73% to 92%, where 4 are between $70\% - 79\%$, 14 are between $80\% - 89\%$ and 3 are between $90\% - 92\%$.

It should be noticed that the table only mentions the number of Temporal-Association-Causal (TAC) rules, not the actions in PDDL domain model, as both are the same. The acquired knowledge from individual datasets has shown reasonable accuracy. Different kinds of plan solutions present varying amount of knowledge for a given security problem. The amount of accuracy depicts that how much of the plan solution was correctly able to resolve the problem. There is potential for a degree of uncertainty in finding the accuracy as the human expert might not know the exact, complete set of events that are produced when a particular security configuration activity is performed on a machine. Moreover, there can be one or more alternative ways of executing the same task, which might produce a different set of event entries. This can lead the expert to make erroneous judgement regarding the validity of sequences of events. There is also uncertainty around the security plan solution as LPG is a sub-optimal planner; however, its ability to apply stochastic local search to draw plan without any goal state motivated the use.

Furthermore, any successfully found plan, optimal or otherwise, will provide one of the many ways to conduct a certain security configuration.

Event dataset	**Number of unique events (total entries)**	**Automated domain modelling time**	**Manual domain modelling time**
1	29 (5,027)	00h:08m:03s	00h:17m:54s
2	31 (8,253)	00h:04m:46s	00h:20m:40s
3	16 (521)	00h:00m:18s	00h:08m:53s
4	20 (1,122)	00h:03m:21s	00h:11m:54s
5	23 (1,079)	00h:01m:09s	00h:13m:27s
6	27 (1,691)	00h:02m:58s	00h:17m:18s
7	25 (1,714)	00h:02m:53s	00h:16m:40s
8	31 (9,721)	00h:04m:34s	00h:22m:28s
9	29 (9,770)	00h:04m:37s	00h:17m:15s
10	29 (13,026)	00h:07m:02s	00h:21m:58s
11	13 (10,948)	00h:05m:29s	00h:14m:26s
12	19 (8,832)	00h:05m:05s	00h:13m:46s
13	32 (3,403)	00h:07m:00s	00h:21m:20s
14	14 (31,719)	00h:26m:04s	00h:29m:10s
15	18 (1,072)	00h:00m:47s	00h:11m:06s
16	33 (1,991)	00h:01m:42s	00h:18m:20s
17	23 (826)	00h:05m:01s	00h:12m:46s
18	29 (5,309)	00h:03m:10s	00h:16m:54s
19	67 (69,854)	00h:38m:33s	01h:05m:41s
20	44 (313,470)	00h:30m:45s	02h:26m:41s
21	21 (32,688)	00h:27m:51s	00h:38m:53s

TABLE 5.3: Comparison of automated and manual domain action models with respect to knowledge acquisition and representation time. The 'Automated domain modelling time' is the execution time of developed solution producing an automated domain model, whilst, the 'Manual domain modelling time' is the time taken by three human experts for creating a manual domain model from the event log dataset. The time is expressed in 'h' for hours, 'm' for minutes and 's' for seconds.

Table 5.3 presents the amount of time required to generate automated (generated by proposed solution) and manual (created by human expert) domain actions models. For automated domain model, time depicts the execution period of a complete processing cycle; reading the event log dataset, building event relationships and writing results in a PDDL file. For manual domain model, the time only depicts the duration of creating event relationships. The table mentions both total and unique number of events in a dataset as these factors highly influence the increase or decrease in the processing time. Following specifications are gathered from the table by comparing the processing time of automated and manual approaches:

1. The maximum automated domain modelling time, 38 minutes and 33 seconds, was taken by dataset-19 to process 69,854 entries having 67 unique events. The same dataset took 1 hours, 5 minutes and 41 seconds for manual processing;

2. The maximum manual domain modelling time, 2 hours, 26 minutes and 41 seconds, was taken by dataset-20 to process 313,470 entries having 44 unique events. The same dataset took 30 minutes and 45 seconds for automated processing;

3. The minimum automated domain modelling time, 47 seconds, was taken by dataset-15 to process 1,079 entries having 23 unique events. The same dataset took 11 minutes and 6 seconds for manual processing; and

4. The minimum manual domain modelling time, 8 minutes and 53 seconds, was taken by dataset-3 to process 521 entries having 16 unique events. The same dataset took 18 seconds for automated processing.

5.2.5 Discussion

This section presents the key findings and interpretation of the evaluation results along with the demonstration of their significance.

5.2.5.1 Completeness

Consider the following enumeration of datasets from Tables 5.2 and 5.3: the dataset-16 has 1,991 total and 33 unique entries; the dataset-19 has 69,854 total and 67 unique entries; and the dataset-20 has 313,470 total and 44 unique entries. The dataset-16 generated the highest number of Temporal-Association-Causal (TAC) rules (75) and the corresponding actions (82) in the plan solutions; although, it is dataset-20 that contains the highest number of event entries out of others. 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 9 actions in the respective plan solution. This indicates that the dataset-11 has relatively large number of routine events that occurred frequently over time, but they were removed during the initial stages of processing, thus reducing the number of final domain actions.

The higher ratio of unique to total event log entries implies substantial user activity and fewer routine/repetitive tasks. As expected in this case, the developed solution produces a larger object-based model, and therefore results in more number of TAC rules or domain actions. The relationship between the number of unique events and domain actions is directly proportional, i.e. higher number of unique events in a dataset produces higher number of domain actions, which demonstrates the solution's ability to extract complete knowledge. This enables the planner to always find a plan for any given security problem, assuming the targeted knowledge is present in the domain model. Therefore, we claim that the proposed solution is fully data-driven and complete.

5.2.5.2 Efficiency

Consider the following observations from Table 5.2: the maximum number of association rules (516,608) is produced by dataset-1 that contains 5,027 event log entries, and resulted in 32 final TAC rules; the dataset-6 generated 263,800 association rules from 1,691 event entries, but only resulted in 6 final TAC rules; the dataset-13 contains 3,403 event entries, which produced 97,574 association rules and only 6 final TAC rules; and the dataset-16 only used 4,959 association rules to output 75 TAC rules from 1,991 event entries. Based on this inconsistent pattern, we can deduce that the number of object-based association rules, TAC rules/domain actions and event log entries are all independent of each other. The output of proposed solution depends on the content and characteristics of input data, for example, the amount of security-related activities stored in the dataset, quantity of routine events and number of distinct object-value pairs. Furthermore, due to the temporal validation of association rules and formation of Partial Ancestral Graph (PAG) of event relationships, the majority of false positive and conflicting rules are either corrected or removed. This signifies the efficiency of developed solution that is achieved by conducting both qualitative and quantitative analysis of input event log entries.

5.2.5.3 Performance

The performance is measured as a time difference between creating a manual and automated domain action model from the same event log dataset. The results presented in Table 5.3 clearly demonstrate that the automated solution is reasonably faster, efficient

and scalable than the manual approach. The automated domain modelling of every dataset took less amount of time than manual domain modelling. Besides, unlike manual domain modelling, the automated solution does not require any human resources, hence making it more suitable for the real time environments. Furthermore, the developed solution eliminates incorrect and irrelevant rules during each phase, which also contributes towards the reduction of overall processing time.

Regarding the manual processing, dataset-10 took 21 minutes and 58 seconds to identify event relationships within 13,026 entries, whilst, dataset-13 took 21 minutes and 20 seconds for 3,403 entries. There is a difference of 9,623 entries between the two datasets; however, the time variation is only 38 seconds. The reason behind this apparent discrepancy is the (approximately similar) amount of unique events, which is 29 and 32 for dataset-10 and dataset-13, respectively. This means dataset-10 contains a large and continuous segment of routine event entries, possibly due to a lengthy user/application/system inactivity, as compared to dataset-13. Such events were quickly skipped by the human experts, thus reducing the manual processing time. Similarly, it took 14 minutes and 26 seconds to extract knowledge from 10,948 entries of dataset-11, whereas for dataset-17, which has 826 entries, consumed 12 minutes and 46 seconds. So, a difference of 10,122 entries only added 1 minute and 41 seconds in time, which is again due to a large portion of routine events.

5.2.5.4 Importance Of Causal Inference

Another important aspect of the evaluation results is the difference between the number of Temporal-Association (TA) and Temporal-Association-Causal (TAC) rules. The TAC rules are always less or sometimes equal in number than the TA rules. The main reason behind this difference is the formation of Directed Acyclic Graph (DAG), which eliminates all redundancies, conflicts and cycles from the TA rules, therefore improving the quality of results. It also depends on the characteristics of event log dataset and the extracted TA rules. For example, the dataset-1 has 84 TA and 32 TAC rules, which means 52 event relationships were removed. Similarly, for dataset-2, 8 TA rules were eliminated. In case of dataset-4, 6, 18 and 21, the number of TA and TAC rules are equal, which shows that correlation can imply causality sometimes; however, it not always true.

Furthermore, calculating and assigning a causal rank to each TAC rule either increases or decreases its overall accumulative-weight value. So, besides temporal-association accuracy and counter values, the causal rank becomes an additional measure that enforces the planner to select the most feasible plan of actions (from entire domain knowledge) to solve a given security issue. This signifies the importance of application of causality that lead towards better accuracy.

5.2.5.5 Other Findings

It has also been observed that the datasets with comparatively lower accuracy consists of such kind of rules, where a single event is connected to several other events simultaneously. To minimise this issue, Automated Planning plays an important role to determine the most feasible set of actions by maximising the accumulative-weight values. Hence, the planner always selects those actions, which will result in the maximum overall accumulative-weight value.

Another observation is that, sometimes, a dataset contains a large number of event log entries, but they are only comprised of few unique event types. Such datasets should not produce many actions in their respective domain models. It is clear from Table 5.2 that the solution does not force event relationships, where they are not present, regardless of number of entries in a dataset. Some of the datasets were excluded during the evaluation phase as they did not produce any domain actions at all. The object-based event associations ensure that a rule, and consequently a domain action, is created only where there exists a relationship among the properties of event log entries.

Another observation is that the automated solution has not shown full accuracy; the maximum accuracy achieved is 92%. Although, the manual approach is capable of providing a fully accurate domain action model from event logs, there still can be errors considering the tiresome as well as continuous creation, updation and alteration of security knowledge (as the process is conducted by human beings). Moreover, there is also a possibility of incomplete knowledge acquisition and representation due to missing one or more relationships.

Apart from the results, it is important to acknowledge that the solution will provide better accuracy if the configuration, environment and intended use of the vulnerable

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 5.4: Automated and manual plans to solve same problem instance. The conversion of plans in a user-friendly format is performed by the plan parser that shows actions and object-value pairs.

machine are same as of the machine from which the domain action model was extracted. The similar settings and context/properties of events will improve the compatibility of learnt security knowledge and help the users, in a better and easier way, to perform the necessary changes.

5.2.6 Accuracy Analysis Of An Automated Plan

This section presents a comparison between an automated and manual plan that showcases the ability of the proposed solution to suggest human-like actions. Both plans are complete, which means they include valid sets of actions for security identification and mitigation. The plans are shown in Table 5.4, whereas a detailed elaboration of the automated plan actions is provided in Section 5.2.7. As mentioned before, the actions of a plan might be difficult for the non-expert to understand, so we used the plan parsing application to elaborate all related events and parameters. The accuracy calculation mechanism covers all aspects for any given plan solution by finding the number of correct, incorrect, additional and missing actions. For the current automated plan, all 8 actions are correct as they are found in the manual plan, and there are no incorrect and additional actions. However, the automated plan is not entirely accurate as it is missing three actions (5, 6 and 7) that are present in the manual plan. The missing

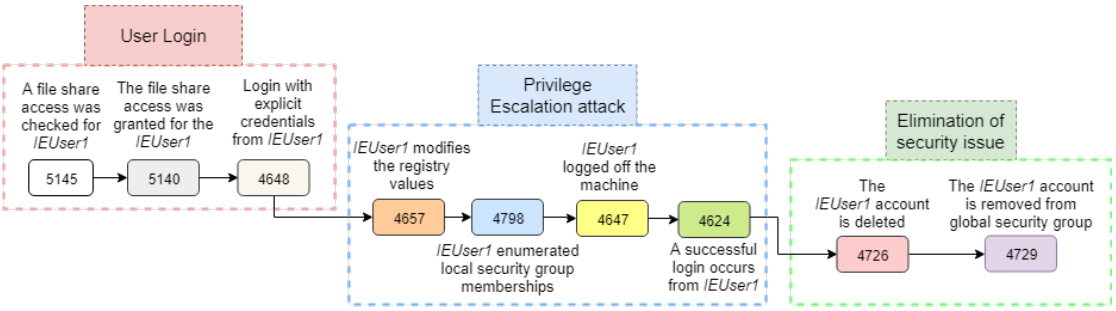

FIGURE 5.9: A complete timeline of events extracted in the form of an automated plan. The related events and corresponding objects reflect the security-related actions of human expert on a particular machine. The red and blue coloured boxes show the events found in vulnerable machine, whilst, the green coloured box shows the mitigation plan.

actions are: 4798 → 4728, 4728 → 4738 and 4738 → 4647. Hence the accuracy would be: $(8/(8+0+3+0)) \times 100 = 73\%$.

5.2.7 Interpretation Of The Plan For Usability

The domain action models are the representatives of TAC rules that vary on the basis of scenarios they are captured in, and how the proposed solution assigns accumulative-weight in the actions. The event entries are logged as soon as they are generated by multiple system activities running in parallel, and there is no way of knowing which of the events are linked corresponding to a single human expert action. So, the solution provided in this research can be used to find those sequences and connections in the presence of large number of routine event entries, and facilitate the autonomous replication of human expert actions.

A timeline demonstrating the automated mitigation plan for a security concern (presented in Table 5.4) is shown in Figure 5.9. The timeline consists of a complete list of actions found in the plan; however, it is important to mention here that a large portion of the actions have already been discovered in the event log of a vulnerable machine and represented in the corresponding problem instance. This section also demonstrates that the proposed solution is easy-to-use as it does not require any human aid. It also shows that an automated plan only requires a fundamental knowledge of computing to interpret and implement the actions. It should be noticed here that the sequences of events generated from a single dataset are not universally applicable, they are subjective

and based on the expert activities performed on the underlying machine. Following is the brief interpretation of automated planner output:

- ***Matched*** – This part of the plan determines 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 that is 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
  - *Privilege Escalation attack* – Following the login, event 4657 is triggered to show that the `IEUser1` added a new registry value. After that, event 4798 informs that the `IEUser1` probed all 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 third-party local application.
- ***Partially Matched*** – This part of the plan provides the security solution, which was discovered by the planner based on 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 in Figure 5.9, it is shown that an administrator found a suspicious activity in terms of system registry modification by a local user account (possibly malicious insider), which lead to the discovery of a privilege escalation attack on a remote server. Initially, it is unknown whether the user account actually became a legitimised member of another user-group, but it became apparent later on 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 to login and added a registry value, which allowed the scanning and identification of other existing users. This process enabled the user to find and change to a user-group with higher access permissions. Furthermore, the plan shows that the newly added value in the registry was a string, which is done through a 'net' command to add the current user in an administrator group, and placed in registry path *HKLM\Software\Microsoft\Windows\currentversion\run*. As the user attempted (and failed) to access the file-share service of the server, the new user-group provisioned sufficient rights to gain unauthorised access to the service. The administrator resolved this security issue by completely removing the malicious user account from the server to prevent any future issues.

Continuing the discussion from Section 5.2.6, the automated plan is missing information as it does not fully match (only 73%) with the manual plan. It depends on the amount of absent event relationships and level of expertise of the user to determine the severity of the missing information in an automated plan solution. As evident by this sequence (4798 $\rightarrow$ 4728 $\rightarrow$ 4738 $\rightarrow$ 4647), after gathering the user-groups (event 4798), the account of `IEUser1` was added to a security-enabled global group (event 4728). This change in the user account `IEUser1` is depicted by the event 4738. Later on, `IEUser1` logged off from the server that triggered the event 4647.

5.2.8 Example Of An Incorrect Automated Plan

Based on our analysis, there is also a potential of generating an incorrect plan solution, where the extracted sequence of actions present inaccurate method of conducting security-related activities. This happens due to the erroneous extraction of event relationships. An example of incorrect plan is presented in Figure 5.10. The object names of the plan are extracted by the plan parser application and explained in the following: `RSA` is the algorithm name, `d5_0e7b98a02239` represents the address of cryptographic-key file path, `SMS` is the key name, `0_2499` is the key type, `AD` is the subject's domain name, `0_0x3e7` login ID of subject, `CES000963029$` is the subject username, `S_1_5_18` is the SID, `0_16390` presents a specific cryptographic-related action, `GUID_MFE_TCP_STREAM_CALLOUT_V4` is the callout function name of Windows Filtering Platform (WFP), `DataFinder` is the filter name of WFP, `Kerberos` is an authentication

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

FIGURE 5.10: Incorrect sequence of actions extracted for the test machine using the acquired domain knowledge.

protocol and `S_1_0_0` is the SID of target user. 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 system startup and does not indicate any configuration change. The last event 4624 indicates that a user-account failed to login 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 in a single sequence. The order of events is also incorrect as all cryptographic operations are performed after the user is logged onto the machine. Furthermore, these actions do not provide any useful or actionable knowledge to the non-experts that might improve their systems' security.

5.2.9 A different approach to evaluation

The automated knowledge acquisition (KA) is performed by the developed solution, whereas, the manual KA is conducted by the human experts as mentioned in Section 5.2.3.4. Both kinds of expert security knowledge are acquired in the form of Temporal-Association-Causal (TAC) rules. In the following discussion, the TAC rules acquired through developed solution are referred as 'automated TAC rules', whereas, the TAC rules acquired from human experts are referred as 'manual TAC rules'. As the human involvement terminates after the 'Manual extraction of event relationships' stage in the evaluation process shown in Figure 5.7, a direct comparison between the automated and manual TAC rules will help in evaluating the acquired knowledge itself, rather than the utilisation of the acquired knowledge. In other words, this comparison

will determine how accurately the developed solution identified patterns of security-related actions in a given event log dataset, instead of how accurately the developed solution suggested the human expert-like actions in resolving a certain security threat or an issue (as described in Section 5.2.3.5). Therefore, this comparison can provide another perspective regarding the accuracy of knowledge acquisition in the developed solution.

Event dataset	**TAC rules accuracy**
1	84%
2	83%
3	75%
4	75%
5	90%
6	83%
7	90%
8	78%
9	91%
10	86%
11	81%
12	81%
13	75%
14	80%
15	83%
16	97%
17	83%
18	80%
19	93%
20	89%
21	86%

TABLE 5.5: Evaluation of knowledge acquisition phase in terms of accuracy using a different approach, i.e. direct comparison between manually and automatically acquired TAC rules.

The results of comparing automated and manual TAC rules from each dataset are shown in Table 5.5. The amount of accuracy ranges from 73% to 97%, where 4 are between $70\% - 79\%$, 12 are between $80\% - 89\%$ and 5 are between $90\% - 92\%$. These results are calculated using the same accuracy formula presented in Equation 5.1. An automated TAC rule would be considered as 'correct' if the same relationship, having identical events and direction, is found in the set of manual TAC rules as well. An automated TAC rule would be considered as 'incorrect' if a rule with the same events is discovered in the set of manual TAC rules, but with opposite relationship direction. An automated

TAC rule would be considered as 'missing' if it exists in the set of manual TAC rules, but is absent from the set of automated TAC rules. An automated TAC rule would be considered as 'additional' if it exists in the set of automated TAC rules but is not found in the set of manual TAC rules.

Using Table 5.2 and Table 5.5, it is visible that for 12 datasets, the accuracy of TAC rules is greater than the accuracy of its corresponding automated plan, whereas, for remaining 9 datasets, the accuracy of automated plan is greater than the accuracy of its respective TAC rules. This inconsistency has occurred mainly due to the following reasons. First, for the given problem, only those actions were found applicable from the domain model that had relatively lower/higher accuracy than other actions. In other words, as the action plan usually presents a subset of the domain model (which is the representation of all TAC rules), it is possible that depending on the problem instance, only those specific actions were included in the plan that had comparatively lower/higher accuracy. Second, the LPG, which is a sub-optimal and stochastic algorithm, randomly selected low/high priority actions. The priority is defined based on the accumulative-weight value. If lower or higher priority actions were chosen, it would result in a decreased or increased accuracy of the automated plan, respectively. Third, the empty goal state forces the planner to choose a plan (albeit valid); however it might not be the best possible plan.

5.3 Limitations

The research work presented in this thesis is novel and demonstrates a reasonably accurate approach of autonomous elicitation and utilisation of security knowledge by processing existing event log sources. However, the approach is a first step in this direction and has the following limitations:

1. The associating rule mining is a compute-intensive task as it is an iterative approach and processes all possible itemset. This significantly decreases the overall efficiency of the developed solution. On the other hand, applying a mechanism that reduces the number of iterations and candidate itemsets for relatively quicker processing, will decrease the quality of association rules;

2. The proposed solution is not capable of modelling complex scenarios, where consecutive repetition of an event is part of the sequence of actions. For example, if an event dataset contains 'suspend the user account for more than three failed log-on attempts' knowledge, it cannot be extracted as an automated rule and represented in PDDL as the proposed solution is not capable of correlating events with themselves;

3. Another issue with the developed solution is that it cannot distinguish between a correct and incorrect plan solution against a given problem instance. In other words, there is no automated mechanism of providing an accuracy value for the generated plan. If a user, especially non-expert, is unaware of a plan being fully or partially incorrect, it can mislead him/her into performing wrong security assessment or configuration;

4. The generalisation of modelling security knowledge has restricted the solution to use domain-independent planners. However, the range of planners can be increased by introducing human assistance in the domain designing and modelling process, which also might result in plans that do not represent the real-world security issues and solutions.

5. The environment, type and context of problem instance are unavailable, whilst, the planner is searching for a solution in the domain action model. This uncertainty can lead to a plan solution, which is not fully applicable to the identified vulnerabilities of the underlying machine.

5.4 Chapter Summary

The purpose of this chapter is to present the implementation and evaluation process of the proposed solution. The implementation consists of two applications. The first application is console-based that extracts the knowledge from event log datasets, stores it in a database and represents in a PDDL-based domain action model. This application can process one or more datasets in a single execution and output both separate and combined domain action models. The second application has a graphical user interface and is capable of applying the domain action model on any vulnerable/unconfigured machine to strengthen its security. This application is integrated with the LPG planner

as well as a plan validator. Both applications are designed simply and efficiently so that they can be used by both expert and non-expert users.

The evaluation of the developed solution includes accuracy and performance analysis. It is performed with the help of three human experts. The event log datasets for evaluation are obtained from 21 live machines, which were configured by experts according to different security policies. The domain action models are produced from each dataset using manual and automated approaches. After that, their comparison is performed to demonstrate: if the automated approach generated a similar domain action model as of human experts (accuracy) and how much time was saved by using the automated tool (performance). The results have shown suitable accuracy, i.e. between 73% and 92%, and a significant reduction of time for every dataset. At the end, this chapter provides a real-world scenario, where the automated tool identified a security issue and presented an acceptable solution. The scenario also helps in demonstrating the usage of the developed tool.

Chapter 6

Conclusion and Future work

6.1 Conclusion

The Chapter 2 of this thesis indicates that there is no existing method to automatically extract security knowledge from a machine and apply it to other machines for strengthening their security. Although a good amount of previous research has been done in the construction of domain action models and automated planning (AP) for security applications, it has two major problems: manual knowledge modelling and attack planning as the only area of focus. The manual domain designing process is quite resource-intensive and a big challenge for human operator as it is not efficient, and requires intimate knowledge of the area. The automated knowledge extraction and modelling of event relationships is a complex task, so is the manual process. Furthermore, we argue and demonstrate that the AP can play an important role in the deliberation of security knowledge as well. Therefore, in this thesis, we present a novel technique that automatically extracts a domain action model from the security event logs of a system, without any assistance from a human operator. The main purpose of this technique is to allow non-expert users to perform expert security analysis of the new or previously unseen (vulnerable) machines without investing significant amount of effort, cost and time in acquiring the required knowledge. This technique provides a computer-based alternative of a scenario, where a human expert performs a security assessment or configuration of a machine.

As mentioned in Section 1.2, all system changes and actions are explicitly recorded by the event logging mechanism. Learning is a fundamental aspect of autonomous behaviour and it can be defined in many different ways. Initially, several techniques were considered for automated learning, such as using clustering algorithm to group events against security actions, developing assistive tools for human experts to produce security templates and applying deep learning algorithms to the text of security-based academic papers and online forums to find security problem-solution pairs. Upon careful examination, these techniques were found unsuitable for the real-time environments due to lack of decision-making criteria and intelligent reasoning for probing security issues and suggesting remedial actions. However, the experimentation by constructing manual domain model using event logs showed promising results in terms of accuracy and performance. So, it was identified that the most suitable method is to adopt the state-of-the-art AP technology, where the knowledge is acquired and encoded using a domain-independent language and utilised using an increased range planning algorithms. Hence, learning domain models for AP is what motivated this research, and a technique was developed to extract temporal-association-causal (TAC) relationships from the event log entries and encode them into PDDL-based domain action model. After that, a planning algorithm utilises the domain model to output one or more sequences of actions, which were performed by the human expert user to identify and mitigate the particular vulnerabilities of the machine. In other words, the proposed technique orchestrates the security evaluation of a system, in a similar way, as a human expert would.

The main contributions of this research work are presented in the following:

1. An automated procedure to determine and eliminate noise from the event logs;
2. A new scheme to find support for association rule mining algorithms;
3. A novel algorithm to create ordered sequences of correlated events based on a temporality metric;
4. A method to define and rank strength of causality within a graph of connected events; and
5. An intelligent mechanism to utilise knowledge on new or previous unknown machines.

6.1.1 Summary

The proposed solution is split into two parts; automated rule mining from event logs to gain and store knowledge (presented in Chapter 3) and intelligent application of the knowledge on vulnerable machines using planning algorithms for security improvements (presented in Chapter 4).

For the rule mining part, the first step is to read all security event log entries of a given system. After that, determine routine events in the given dataset and eliminate them. Next, the solution creates an object-based model to represent event log entries in a unified format, and determines minimum and maximum support values (referred as a support range in the thesis) for the association rule mining (ARM) algorithm. In the next step, the solution mines strong object-based correlation rules using a modified Apriori algorithm. The object-based rules are converted into the event-based rules where any two event types are associated by the object-based rules. The quality of event relationships is improved by ordering the events based on a temporal metric, whilst, filtering all insignificant relationships where the temporal value is below threshold. After that, the event-based correlation rules are combined and validated into sequences of temporally-associated rules. These rules are further converted into a directed acyclic graph and assigned a causal rank using the FCI algorithm, hence obtaining temporal-association-causal (TAC) rules. All TAC are stored into MySQL database as well for future use.

For automated planning part, the knowledge extracted in the form of TAC rules is encoded into a stand-alone domain action model using PDDL version 2.1. The PDDL format provides a standardised mechanism of representing and storing the extracted knowledge, along with fulfilling the required level of expressiveness to support domains. It also increases the adoption of the solution due to wider understanding and presence of domain independent automated planners. Apart from domain model, the developed solution is also capable of automatically generating problem instances by processing live events and their objects from any (vulnerable) machine. Given the domain and problem files, LPG planner is used to produce relevant actions in the form of a plan that can be utilised by the non-expert users to enhance the machine's security. The plan is responsible for identifying security issues and providing mitigation actions. The LPG is one of the most powerful domain-independent planners that has the ability to solve problems of a large variety with diverse requirements (numeric fluents, minimisation and

maximisation metrics, etc.). It should be noticed here that the research into domain-independent planners is progressing rapidly, and new planners are being continuously developed. It is envisaged that in the future, there will be an increased variety of planners to solve the current domain model with better performance and accuracy.

Through an extensive empirical analysis, the proposed automated solution has shown to be effective. The technique was implemented as two separate applications; one for knowledge acquisition and representation, and other for knowledge utilisation. The developed solution was evaluated on live, real-world event log datasets to determine the accuracy, completeness and performance measures. The solution achieved $73\% - 92\%$ accuracy for 21 datasets and a significant performance increase against the manual approach. Furthermore, a detailed comparison between an automated and manual plan solution has been performed to demonstrate the autonomous replication of human expert actions along with the ease of usability and applicability in real time environments. The event log entries are generally produced in high frequency, but a significant portion of them represents routine events (noise). Despite this, the proposed solution has clearly demonstrated that it is capable of extracting the domain knowledge with a reasonable accuracy and be applicable in practical environments, meanwhile, reducing manual time and effort and financial cost without the assistance from a human expert.

6.2 Suggested Future Work

The research work presented in this thesis several potential areas for research. Moreover, there are several avenues that can be explored for future development of this solution. Some are summarised below based on their importance:

1. To apply the proposed technique on other operating systems and applications, there is a need to create parsing tools or software modules that can process the respective event log datasets to generate object-based models. The event log entries are found in different text-formats, and requires different parsing and analysis mechanisms to obtain objects (as described in Section 3.1.2). Doing this will increase the applicability and adoption of the proposed solution;

2. There are some event logging mechanisms, e.g. application-specific security event logs, that do not assign a numeric identifier to an event. In general, it is considered a good practice to add a unique identifier for every event type as it increases the parsing accuracy [197]. A small amount of research has been done that automatically introduces event identifiers in the raw event entries [198]. Extending and incorporating such software modules in the developed solution will also lead towards wider adoption;

3. The next stage of this research is the augmentation of plan parser tool, which is described in Section 5.1.2.2. The would include the development of automated tools and techniques, which are capable of translating any valid plan solution into concrete set of actions. After that, the actions will be converted into (a) system commands (where possible) and (b) complete step-by-step guideline that can be executed on the vulnerable machine. This will remove or reduce the manual effort of non-expert users implementing the recommended security measures;

4. From the utilisation perspective, the developed solution described in Section 5.1 can be expanded into a web-application that can be easily used by numerous individuals/organisations, simultaneously. A large and diverse set of event logs from multiple sources would create a bigger, central repository of TAC rules. This would allow the domain action model to contain more and better knowledge for identifying security issues and suggesting solutions; and

5. As highlighted in Section 4.3 and Section 4.3.4.1, PDDL-based representation of knowledge and LPG planning algorithm, respectively, have shown great potential to solve large and complicated planning problems. This motivates the further advancement of tools and languages for knowledge engineering, so that they can be utilised in commercial settings. The improvements will increase the efficiency, quality and performance of plan generation along with benefiting all application areas of automated planning.

References

[1] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: theory and practice*. Elsevier, 2004.

[2] Valentina Viduto, Carsten Maple, Wei Huang, and David LóPez-PeréZ. A novel risk assessment and o model for a multi-objective network security countermeasure selection problem. *Decision Support Systems*, 53(3):599–610, 2012.

[3] Leonardo Mariani, Mauro Pezzè, and Mauro Santoro. Gk-tail+ an efficient approach to learn software models. *IEEE Transactions on Software Engineering*, 43 (8):715–738, 2017.

[4] Tien-Duy B Le and David Lo. Deep specification mining. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 106–117. ACM, 2018.

[5] Tien-Duy B Le, Xuan-Bach D Le, David Lo, and Ivan Beschastnikh. Synergizing specification miners through model fissions and fusions (t). In *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, pages 115–125. IEEE, 2015.

[6] Andrzej Wasylkowski and Andreas Zeller. Mining temporal specifications from object usage. *Automated Software Engineering*, 18(3-4):263–292, 2011.

[7] Derek Schauland and Donald Jacobs. Managing the windows event log. In *Troubleshooting Windows Server with PowerShell*, pages 17–33. Springer, 2016.

[8] Cristina Simache, Mohamed Kaâniche, and Ayda Saidane. Event log based dependability analysis of windows nt and 2k systems. In *Dependable Computing, 2002. Proceedings. 2002 Pacific Rim International Symposium on*, pages 311–315. IEEE, 2002.

[9] Muhammad Faheem Mushtaq, Urooj Akram, Irfan Khan, Sundas Naqeeb Khan, Asim Shahzad, and Arif Ullah. Cloud computing environment and security challenges: A review. *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, 8(10):183–195, 2017.

[10] Saad Khan, Simon Parkinson, and Yongrui Qin. Fog computing security: a review of current applications and security solutions. *Journal of Cloud Computing*, 6(1): 19, Aug 2017. ISSN 2192-113X. doi: 10.1186/s13677-017-0090-3. URL https://doi.org/10.1186/s13677-017-0090-3.

[11] David 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*, pages 3–3. Springer, 2008.

[12] Changwei Liu, Anoop Singhal, and Duminda Wijesekera. A model towards using evidence from security events for network attack analysis. In *WOSIS*, pages 83–95, 2014.

[13] Risto Vaarandi. Mining event logs with slct and loghound. In *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pages 1071–1074. IEEE, 2008.

[14] Justin Myers, Michael R Grimaila, and Robert F Mills. Log-based distributed security event detection using simple event correlator. In *System Sciences (HICSS), 2011 44th Hawaii International Conference on*, pages 1–7. IEEE, 2011.

[15] Judea Pearl et al. Causal inference in statistics: An overview. *Statistics surveys*, 3:96–146, 2009.

[16] Clemens Sauerwein, Irdin Pekaric, Michael Felderer, and Ruth Breu. An analysis and classification of public information security data sources used in research and practice. *Computers & Security*, 82:140–155, 2019.

[17] Peter Holdt Christensen. Knowledge sharing: moving away from the obsession with best practices. *Journal of knowledge management*, 11(1):36–47, 2007.

[18] Thomas R Gruber. Automated knowledge acquisition for strategic knowledge. In *Knowledge Acquisition: Selected Research and Commentary*, pages 47–90. Springer, 1989.

[19] Tu Bao Ho, Saori Kawasaki, and Janusz Granat. Knowledge acquisition by machine learning and data mining. In *Creative Environments*, pages 69–91. Springer, 2007.

[20] Amit Agarwal, David Ahrens, Rod Livingood, Mahalingam Mani, Navjot Singh, and Andrew Zmolek. Multi-tier security event correlation and mitigation, October 8 2009. US Patent App. 12/234,248.

[21] John P Rouillard. Real-time log file analysis using the simple event correlator (sec). In *LISA*, volume 4, pages 133–150, 2004.

[22] James E Prewett. Analyzing cluster log files using logsurfer. In *Proceedings of the 4th Annual Conference on Linux Clusters*. Citeseer, 2003.

[23] Paul Krizak. Log analysis and event correlation using variable temporal event correlator (vtec). In *LISA*, 2010.

[24] Manuel Peña, Félix Biscarri, Juan Ignacio Guerrero, Iñigo Monedero, and Carlos León. Rule-based system to detect energy efficiency anomalies in smart buildings, a data mining approach. *Expert Systems with Applications*, 56:242–255, 2016.

[25] Nada Sharaf, Slim Abdennadher, and Thom Frühwirth. A rule-based approach for animating java algorithms. In *2016 20th International Conference Information Visualisation (IV)*, pages 141–145. IEEE, 2016.

[26] Mark Devaney, Ashwin Ram, Hai Qiu, and Jay Lee. Preventing failures by mining maintenance logs with case-based reasoning. In *Proceedings of the 59th meeting of the society for machinery failure prevention technology (MFPT-59)*, 2005.

[27] Stylianos Kapetanakis, Miltiadis Petridis, Jixin Ma, and Liz Bacon. Workflow monitoring and diagnosis using case based reasoning on incomplete temporal log data. In *Proceedings of the 8th International Conference on Case Based Reasoning, Seattle, USA*. University of Brighton, 2009.

[28] Stelios Kapetanakis, Avgoustinos Filippoupolitis, George Loukas, and Tariq Saad Al Murayziq. Profiling cyber attackers using case-based reasoning. 2014.

[29] Robert F Erbacher and Steve E Hutchinson. Extending case-based reasoning to network alert reporting. In *2012 International Conference on Cyber Security*, pages 187–194. IEEE, 2012.

[30] Padraig Cunningham. Cbr: Strengths and weaknesses. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 517–524. Springer, 1998.

[31] Shaula Alexander Yemini, Shmuel Kliger, Eyal Mozes, Yechiam Yemini, and David Ohsie. High speed and robust event correlation. *IEEE communications Magazine*, 34(5):82–90, 1996.

[32] Jun Zheng and Mingzeng Hu. An anomaly intrusion detection system based on vector quantization. *IEICE transactions on information and systems*, 89(1):201–210, 2006.

[33] Fabien Pouget and Marc Dacier. Alert correlation: Review of the state of the art. *TechnicalReport EURECOM*, 1271, 2003.

[34] Xiaojiang Du, Mark A Shayman, and Ronald A Skoog. Using neural network in distributed management to identify control and management plane poison messages. In *IEEE Military Communications Conference, 2003. MILCOM 2003.*, volume 1, pages 458–463. IEEE, 2003.

[35] Shraddha S More and Pranit P Gaikwad. Trust-based voting method for efficient malware detection. *Procedia Computer Science*, 79:657–667, 2016.

[36] Plamen P Angelov. Autonomous learning systems: from data streams to knowledge in real-time. 2013.

[37] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

[38] Piyush Rai. Data clustering: K-means and hierarchical clustering. *CS5350/6350: Machine Learning Oct*, 4:24, 2011.

[39] Charu C Aggarwal and Chandan K Reddy. *Data clustering: algorithms and applications*. CRC press, 2013.

[40] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):881–892, 2002.

[41] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[42] Pavel Berkhin et al. A survey of clustering data mining techniques. *Grouping multidimensional data*, 25:71, 2006.

[43] Ruizhi Zhang, Jieren Cheng, Xiangyan Tang, Qiang Liu, and Xiangfeng He. Ddos attack security situation assessment model using fusion feature based on fuzzy c-means clustering algorithm. In *International Conference on Cloud Computing and Security*, pages 654–669. Springer, 2018.

[44] Shi-Jinn Horng, Ming-Yang Su, Yuan-Hsin Chen, Tzong-Wann Kao, Rong-Jian Chen, Jui-Lin Lai, and Citra Dwi Perkasa. A novel intrusion detection system based on hierarchical clustering and support vector machines. *Expert systems with Applications*, 38(1):306–313, 2011.

[45] Risto Vaarandi. A data clustering algorithm for mining patterns from event logs. In *IP Operations & Management, 2003.(IPOM 2003). 3rd IEEE Workshop on*, pages 119–126. IEEE, 2003.

[46] TS Vaquero, Rosimarci Tonaco, Gustavo Costa, Flavio Tonidandel, José Reinaldo Silva, and J Christopher 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)*, pages 11–14, 2012.

[47] Ron M Simpson, T Lee McCluskey, Weihong Zhao, Ruth S Aylett, and Christophe Doniat. Gipo: an integrated graphical tool to support knowledge engineering in ai planning. In *Sixth European Conference on Planning*, 2014.

[48] Javier Barreiro, Matthew Boyce, Minh Do, Jeremy Frank, Michael Iatauro, Tatiana Kichkaylo, Paul Morris, James Ong, Emilio Remolina, Tristan Smith, et al. Europa: a platform for ai planning, scheduling, constraint programming, and optimization. *4th International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS)*, 2012.

[49] Dafna Shahaf and Eyal Amir. Learning partially observable action schemas. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page

913. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[50] Thomas Leo McCluskey, SN Cresswell, N Elisabeth Richardson, and Margaret Mary West. Action knowledge acquisition with opmaker2. In *International Conference on Agents and Artificial Intelligence*, pages 137–150. Springer, 2009.

[51] Stephen N Cresswell, Thomas L McCluskey, and Margaret M West. Acquiring planning domain models using locm. *The Knowledge Engineering Review*, 28(2): 195–213, 2013.

[52] Adi Botea, Markus Enzenberger, Martin Müller, and Jonathan Schaeffer. Macro-ff: Improving ai planning with automatically learned macro-operators. *Journal of Artificial Intelligence Research*, 24:581–621, 2005.

[53] Rabia Jilani, Andrew Crampton, Diane Kitchin, and Mauro Vallati. Ascol: A tool for improving automatic planning domain model acquisition. In *Congress of the Italian Association for Artificial Intelligence*, pages 438–451. Springer, 2015.

[54] Hankz Hankui Zhuo. Crowdsourced action-model acquisition for planning. In *AAAI*, pages 3439–3446, 2015.

[55] Katie Long, Jainarayan Radhakrishnan, Rushabh Shah, and Ashwin Ram. Learning from human demonstrations for real-time case-based planning. 2009.

[56] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[57] Edwin PD Pednault. Formulating multiagent, dynamic-world problems in the classical planning framework. In *Reasoning about actions & plans*, pages 47–82. Elsevier, 1987.

[58] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. Pddl-the planning domain definition language. 1998.

[59] Maria Fox and Derek Long. Pddl2.1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research*, 20:61–124, 2003.

[60] Alfonso Gerevini and Derek Long. Bnf description of pddl3. 0. *Unpublished manuscript from the IPC-5 website*, 2005.

[61] M Helmert. Changes in pddl 3.1. *Unpublished summary from the IPC-2008 website*, 2008.

[62] Håkan LS Younes and Michael L Littman. Ppddl1. 0: An extension to pddl for expressing planning domains with probabilistic effects. *Techn. Rep. CMU-CS-04-162*, 2004.

[63] Maria Fox and Derek Long. Modelling mixed discrete-continuous domains for planning. *Journal of Artificial Intelligence Research*, 27:235–297, 2006.

[64] Jeremy Frank and Ari Jónsson. Constraint-based attribute and interval planning. *Constraints*, 8(4):339–364, 2003.

[65] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[66] Tom Everitt and Marcus Hutter. Analytical results on the bfs vs. dfs algorithm selection problem. part i: tree search. In *Australasian Joint Conference on Artificial Intelligence*, pages 157–165. Springer, 2015.

[67] Jürgen Lerner, Dorothea Wagner, and Katharina Zweig. *Algorithmics of large and complex networks: design, analysis, and simulation*, volume 5515. Springer, 2009.

[68] Mohit Tawarmalani and Nikolaos V Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2):225–249, 2005.

[69] Mauro Vallati, Lukáš Chrpa, Marek Grzes, Thomas L McCluskey, Mark Roberts, and Scott Sanner. The 2014 international planning competition: Progress and trends. *AI Magazine*, 36(3):90–98, 2015.

[70] Saad Khan and Simon Parkinson. Towards automated vulnerability assessment. 2017.

[71] Mark S Boddy, Johnathan Gohde, Thomas Haigh, and Steven A Harp. Course of action generation for cyber security using classical planning. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 12–21, 2005.

[72] Jörg Hoffmann. The metric-ff planning system: Translating"ignoring delete lists"to numeric state variables. *Journal of Artificial Intelligence Research*, 20: 291–341, 2003.

[73] Jorge Lucangeli Obes, Carlos Sarraute, and Gerardo Richarte. Attack planning in the real world. *arXiv preprint arXiv:1306.4044*, 2013.

[74] Anton Riabov, Shirin Sohrabi, Octavian Udrea, and Oktie Hassanzadeh. Efficient high quality plan exploration for network security. In *11th Scheduling and Planning Applications woRKshop(SPARK)*, 2016.

[75] Carlos Sarraute, Gerardo Richarte, and Jorge Lucángeli Obes. An algorithm to find optimal attack paths in nondeterministic scenarios. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 71–80. ACM, 2011.

[76] Carlos Sarraute, Olivier Buffet, and Jörg Hoffmann. Penetration testing== pomdp solving? *arXiv preprint arXiv:1306.4714*, 2013.

[77] Carlos Sarraute, Olivier Buffet, and Jörg Hoffmann. Pomdps make better hackers: Accounting for uncertainty in penetration testing. *arXiv preprint arXiv:1307.8182*, 2013.

[78] Jörg Hoffmann. Simulated penetration testing: From" dijkstra" to" turing test++". In *ICAPS*, pages 364–372, 2015.

[79] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mininga general survey and comparison. *ACM sigkdd explorations newsletter*, 2(1):58–64, 2000.

[80] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Acm sigmod record*, volume 22, pages 207–216. ACM, 1993.

[81] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[82] Markus Hegland. The apriori algorithm–a tutorial. In *Mathematics and computation in imaging science and information processing*, pages 209–262. World Scientific, 2007.

[83] Avigdor Gal. Evaluating matching algorithms: the monotonicity principle. In *Semantic Integration Workshop (SI-2003)*, page 167, 2003.

[84] Jong Soo Park, Ming-Syan Chen, and Philip S Yu. *An effective hash-based algorithm for mining association rules*, volume 24. ACM, 1995.

[85] Zherui Cao, Yuan Tian, Tien-Duy B Le, and David Lo. Rule-based specification mining leveraging learning to rank. *Automated Software Engineering*, pages 1–30, 2018.

[86] Xiaoyu Fu, Rui Ren, Jianfeng Zhan, Wei Zhou, Zhen Jia, and Gang Lu. Logmaster: Mining event correlations in logs of large-scale cluster systems. In *2012 IEEE 31st Symposium on Reliable Distributed Systems*, pages 71–80. IEEE, 2012.

[87] Youcef Djenouri, Asma Belhadi, and Philippe Fournier-Viger. Extracting useful knowledge from event logs: a frequent itemset mining approach. *Knowledge-Based Systems*, 139:132–148, 2018.

[88] S. Parkinson, V. Somaraki, and R. Ward. Auditing file system permissions using association rule mining. *Expert Systems with Applications*, 55:274 – 283, 2016. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2016.02.027. URL http://www.sciencedirect.com/science/article/pii/S0957417416300586.

[89] Chunye Zhao, Shanshan Tu, Haoyu Chen, and Yongfeng Huang. Efficient association rule mining algorithm based on user behavior for cloud security auditing. In *2016 IEEE International Conference of Online Analysis and Computing Science (ICOACS)*, pages 145–149. IEEE, 2016.

[90] S Carolin Jeeva and Elijah Blessing Rajsingh. Intelligent phishing url detection using association rule mining. *Human-centric Computing and Information Sciences*, 6(1):10, 2016.

[91] Tao Ban, Masashi Eto, Shanqing Guo, Daisuke Inoue, Koji Nakao, and Runhe Huang. A study on association rule mining of darknet big data. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2015.

[92] Stephen L Morgan and Christopher Winship. *Counterfactuals and causal inference*. Cambridge University Press, 2014.

[93] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[94] John F Roddick and Myra Spiliopoulou. A survey of temporal knowledge discovery paradigms and methods. *IEEE Transactions on Knowledge and data engineering*, 14(4):750–767, 2002.

[95] Zhai Liang, Tang Xinming, Li Lin, and Jiang 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.

[96] Edi Winarko and John F Roddick. Armada–an algorithm for discovering richer relative temporal association rules from interval-based data. *Data & Knowledge Engineering*, 63(1):76–90, 2007.

[97] Ling Wang, Jianyao Meng, Peipei Xu, and Kaixiang Peng. Mining temporal association rules with frequent itemsets tree. *Applied Soft Computing*, 62:817–829, 2018.

[98] Ting-Feng Tan, Qing-Guo Wang, Tian-He Phang, Xian Li, Jiangshuai Huang, and Dan Zhang. Temporal association rule mining. In *International Conference on Intelligent Science and Big Data Engineering*, pages 247–257. Springer, 2015.

[99] Susanne Bleisch, Matt Duckham, Antony Galton, Patrick Laube, and Jarod Lyon. Mining candidate causal relationships in movement patterns. *International Journal of Geographical Information Science*, 28(2):363–382, 2014.

[100] Andrey Fedorchenko, Igor Kotenko, and Didier 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*, volume 1, pages 270–276. IEEE, 2017.

[101] Ehsan Nazerfard, Parisa Rashidi, and Diane J Cook. Using association rule mining to discover temporal relations of daily activities. In *International Conference On Smart homes and health Telematics*, pages 49–56. Springer, 2011.

[102] John Aldrich et al. Correlations genuine and spurious in pearson and yule. *Statistical science*, 10(4):364–376, 1995.

[103] Karim Chalak and Halbert White. Causality, conditional independence, and graphical separation in settable systems. *Neural Computation*, 24(7):1611–1668, 2012.

[104] Craig Silverstein, Sergey Brin, Rajeev Motwani, and Jeff Ullman. Scalable techniques for mining causal structures. *Data Mining and Knowledge Discovery*, 4 (2-3):163–192, 2000.

[105] David Heckerman, Christopher Meek, and Gregory Cooper. A bayesian approach to causal discovery. In *Innovations in Machine Learning*, pages 1–28. Springer, 2006.

[106] Kui Yu, Jiuyong Li, and Lin Liu. A review on algorithms for constraint-based causal discovery. *arXiv preprint arXiv:1611.03977*, 2016.

[107] Gregory F Cooper. A simple constraint-based algorithm for efficiently mining observational databases for causal relationships. *Data Mining and Knowledge Discovery*, 1(2):203–224, 1997.

[108] Jean-Philippe Pellet and André Elisseeff. Finding latent causes in causal networks: an efficient approach based on markov blankets. In *Advances in Neural Information Processing Systems*, pages 1249–1256, 2009.

[109] David Danks. Learning the causal structure of overlapping variable sets. In *International Conference on Discovery Science*, pages 178–191. Springer, 2002.

[110] Gregory F Cooper and Changwon Yoo. Causal discovery from a mixture of experimental and observational data. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 116–125. Morgan Kaufmann Publishers Inc., 1999.

[111] Subramani Mani and Gregory F Cooper. Causal discovery using a bayesian local causal discovery algorithm. In *Medinfo*, pages 731–735, 2004.

[112] Constantin F Aliferis, Alexander Statnikov, Ioannis Tsamardinos, Subramani Mani, and Xenofon D Koutsoukos. Local causal and markov blanket induction for causal discovery and feature selection for classification part ii: Analysis and extensions. *Journal of Machine Learning Research*, 11(Jan):235–284, 2010.

[113] Jiuyong Li, Thuc Duy Le, Lin Liu, Jixue Liu, Zhou Jin, and Bingyu Sun. Mining causal association rules. In *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, pages 114–123. IEEE, 2013.

[114] Matthew J Rattigan, Marc E Maier, and David D Jensen. Relational blocking for causal discovery. In *AAAI*, 2011.

[115] Holger Schünemann, Suzanne Hill, Gordon Guyatt, Elie A Akl, and Faruque Ahmed. The grade approach and bradford hill's criteria for causation. *Journal of Epidemiology & Community Health*, 65(5):392–395, 2011.

[116] Amy Sliva, Scott Neal Reilly, Randy Casstevens, and John Chamberlain. Tools for validating causal and predictive claims in social science models. *Procedia Manufacturing*, 3:3925–3932, 2015.

[117] Saurav Acharya and Byung Suk Lee. Incremental causal network construction over event streams. *Information Sciences*, 261:32–51, 2014.

[118] J Martin Bland and Douglas G Altman. Statistics notes: measurement error. *Bmj*, 313(7059):744, 1996.

[119] Paul R Wolf and Charles D Ghilani. *Adjustment computations: statistics and least squares in surveying and GIS*. Wiley-Interscience, 1997.

[120] Hervé Abdi. Coefficient of variation. *Encyclopedia of research design*, 1:169–171, 2010.

[121] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766, 2013.

[122] Charu C Aggarwal and ChengXiang Zhai. *Mining text data*. Springer Science & Business Media, 2012.

[123] Johannes Ledolter. *Data mining and business analytics with R*. John Wiley & Sons, 2013.

[124] Kanwal Garg and Deepak Kumar. Comparing the performance of frequent pattern mining algorithms. *International Journal of Computer Applications*, 69(25), 2013.

[125] Hisao Ishibuchi, Isao Kuwajima, and Yusuke Nojima. Prescreening of candidate rules using association rule mining and pareto-optimality in genetic rule selection. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 509–516. Springer, 2007.

[126] Pankaj Kumar Deva Sarma and Anjana Kakati Mahanta. Reduction of number of association rules with inter itemset distance in transaction databases. *International Journal of Database Management Systems*, 4(5):61, 2012.

[127] C Tew, C Giraud-Carrier, K Tanner, and S Burton. Behavior-based clustering and analysis of interestingness measures for association rule mining. *Data Mining and Knowledge Discovery*, 28(4):1004–1045, 2014.

[128] Sunitha Vanamala, L Padma Sree, and S Durga Bhavani. Rare association rule mining for data stream. In *Computer and Communications Technologies (ICCCT), 2014 International Conference on*, pages 1–6. IEEE, 2014.

[129] Wen-Yang Lin and Ming-Cheng Tseng. Automated support specification for efficient mining of interesting association rules. *Journal of Information Science*, 32 (3):238–250, 2006.

[130] CS Kanimozhi Selvi and A Tamilarasi. An automated association rule mining technique with cumulative support thresholds. *Int. J. Open Problems in Compt. Math*, 2(3), 2009.

[131] Dimitar Hristovski, Janez Stare, Borut Peterlin, and Saso Dzeroski. Supporting discovery in medicine by association rule mining in medline and umls. *Studies in health technology and informatics*, (2):1344–1348, 2001.

[132] Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

[133] Samuel Sanford Shapiro and Martin B Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965.

[134] Nornadiah Mohd Razali, Yap Bee Wah, et al. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics*, 2(1):21–33, 2011.

[135] AA Makarov and GI Simonova. Comparative analysis of the powers of the two-sample kolmogorov–smirnov and anderson–darling tests under various alternatives. *Journal of Mathematical Sciences*, pages 1–6, 2018.

[136] Yuanhui Xiao. A fast algorithm for two-dimensional kolmogorov–smirnov two sample tests. *Computational Statistics & Data Analysis*, 105:53–58, 2017.

[137] TW Kirkman. Statistics to use. 1996. *URL: http://www. physics. csbsju. edu/stats/Accessed*, 31:12, 2007.

[138] Aad W Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 1998.

[139] A Makarov and 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.

[140] Srivatsan Laxman and P Shanti Sastry. A survey of temporal data mining. *Sadhana*, 31(2):173–198, 2006.

[141] Rafael S Parpinelli, Heitor S Lopes, and Alex Alves Freitas. Data mining with an ant colony optimization algorithm. *IEEE transactions on evolutionary computation*, 6(4):321–332, 2002.

[142] Biao Qin, Yuni Xia, Sunil Prabhakar, and Yicheng Tu. A rule-based classification algorithm for uncertain data. In *2009 IEEE 25th International Conference on Data Engineering*, pages 1633–1640. IEEE, 2009.

[143] Judea Pearl. *Causality: models, reasoning, and inference*. Cambridge University Press, Cambridge, 2nd edition, 2009. ISBN 9780521773621;052189560X;9780521895606;0521773628;.

[144] Judea Pearl. Causality: models, reasoning, and inference. *Econometric Theory*, 19(675-685):46, 2003.

[145] Jiri Barnat, Lubos Brim, and Petr Rockai. Parallel partial order reduction with topological sort proviso. In *Software Engineering and Formal Methods (SEFM), 2010 8th IEEE International Conference on*, pages 222–231. IEEE, 2010.

[146] Peter Spirtes and Clark Glymour. An algorithm for fast recovery of sparse causal graphs. *Social science computer review*, 9(1):62–72, 1991.

[147] Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, prediction, and search*. Cambridge MA: MIT Press, 2000.

[148] Tom Claassen and Tom Heskes. A logical characterization of constraint-based causal discovery. *arXiv preprint arXiv:1202.3711*, 2012.

[149] Camden Cheek, Huiyong Zheng, Brian R Hallstrom, and Richard E Hughes. Application of a causal discovery algorithm to the analysis of arthroplasty registry data. *Biomedical engineering and computational biology*, 9:1179597218756896, 2018.

[150] Imme Ebert-Uphoff and Yi Deng. Causal discovery for climate research using graphical models. *Journal of Climate*, 25(17):5648–5665, 2012.

[151] Markus Kalisch and Peter Bühlmann. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *Journal of Machine Learning Research*, 8(Mar):613–636, 2007.

[152] Dan Geiger, Thomas Verma, and Judea Pearl. d-separation: From theorems to algorithms. In *Machine Intelligence and Pattern Recognition*, volume 10, pages 139–148. Elsevier, 1990.

[153] Peter Spirtes. Using d-separation to calculate zero partial correlations in linear models with correlated errors. 1996.

[154] Richard Scheines. D-separation. urlhttps://www.andrew.cmu.edu/user/-scheines/tutor/d-sep.html/, 2018. [Online; accessed 07-April-2018].

[155] Doris Entner and Patrik O Hoyer. On causal discovery from time series data using fci. *Probabilistic graphical models*, pages 121–128, 2010.

[156] Patrik O Hoyer, Shohei Shimizu, Antti J Kerminen, and Markus Palviainen. Estimation of causal effects using linear non-gaussian causal models with hidden variables. *International Journal of Approximate Reasoning*, 49(2):362–378, 2008.

[157] Diego Colombo and Marloes H Maathuis. Order-independent constraint-based causal structure learning. *The Journal of Machine Learning Research*, 15(1):3741–3782, 2014.

[158] Jin Tian. Generating markov equivalent maximal ancestral graphs by single edge replacement. *arXiv preprint arXiv:1207.1428*, 2012.

[159] Diego Colombo, Marloes H Maathuis, Markus Kalisch, and Thomas S Richardson. Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, pages 294–321, 2012.

[160] R Ayesha Ali, Thomas S Richardson, Peter Spirtes, et al. Markov equivalence for ancestral graphs. *The Annals of Statistics*, 37(5B):2808–2837, 2009.

[161] Jiji Zhang. A characterization of markov equivalence classes for directed acyclic graphs with latent variables. *arXiv preprint arXiv:1206.5282*, 2012.

[162] P Spirtes, C Meek, and T Richardson. An algorithm for causal inference in the presence of latent variables and selection bias in computation, causation and discovery, 1999, 1999.

[163] Jiji Zhang. On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence*, 172(16-17):1873–1896, 2008.

[164] R Ayesha Ali and Thomas S Richardson. Markov equivalence classes for maximal ancestral graphs. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 1–9. Morgan Kaufmann Publishers Inc., 2002.

[165] Jiji Zhang. Causal reasoning with ancestral graphs. *Journal of Machine Learning Research*, 9(Jul):1437–1474, 2008.

[166] Markus Kalisch, Martin Mächler, Diego Colombo, Marloes H Maathuis, Peter Bühlmann, et al. Causal inference using graphical models with the r package pcalg. *Journal of Statistical Software*, 47(11):1–26, 2012.

[167] Shohei Shimizu, Patrik O Hoyer, Aapo Hyvärinen, and Antti Kerminen. A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7(Oct):2003–2030, 2006.

[168] Christos Bechlivanidis and David A Lagnado. Time reordered: Causal perception guides the interpretation of temporal order. *Cognition*, 146:58–66, 2016.

[169] ALLEN Newell and HA Simon. Gps a program that simulates human thoughts in ea feigenbaum and j. feldman eds., computer and thoughts, 1963.

[170] J. McCarthy and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In Bonnie Lynn Webber and Nils J. Nilsson, editors, *Readings in Artificial Intelligence*, pages 431–450. Morgan Kaufmann, 1981. ISBN 978-0-934613-03-3. doi: https://doi.org/10.1016/B978-0-934613-03-3.50033-7. URL http://www.sciencedirect.com/science/article/pii/B9780934613033500337.

[171] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial intelligence: a modern approach*, volume 74. Prentice hall Englewood Cliffs, 1995.

[172] Austin Tate, Gerhard Wickler, Lee McCluskey, and Lukáš Chrpa. Machine learning and adaptation of domain models to support real time planning in autonomous systems. *HEdLAMP–Huddersfield+ Edinburgh: Learning and Adaptation of Models for Planning, University of Edinburgh*, 2012.

[173] S Shoeeb and T.L. McCluskey. On comparing planning domain models. In *The 29th Workshop of the UK Planning and Scheduling Special Interest Group PlanSIG 2011*, pages 92–94. UK PLANNING AND SCHEDULING Special Interest Group, December 2011. URL http://eprints.hud.ac.uk/id/eprint/12717/.

[174] Arturo González Ferrer. *Knowledge engineering techniques for the translation of process models into temporal hierarchical planning and scheduling domains*. PhD thesis, Ph. D. Dissertation, Universidad de Granada, 2011.

[175] Thomas L McCluskey, Tiago Vaquero, and Mauro Vallati. Issues in planning domain model engineering. 2016.

[176] Susanne Biundo, Ruth Aylett, Michael Beetz, Daniel Borrajo, Amedeo Cesta, Tim Grant, TL McCluskey, Alfredo Milani, and Gerard Verfaille. Technological roadmap on ai planning and scheduling. 2003.

[177] Derek Long and Maria Fox. The 3rd international planning competition: Results and analysis. *Journal of Artificial Intelligence Research*, 20:1–59, 2003.

[178] Pablo Munoz, María D R-Moreno, and Bonifacio Castano. Integrating a pddl-based planner and a plexil-executor into the ptinto robot. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 72–81. Springer, 2010.

[179] Dana S Nau. Current trends in automated planning. *AI magazine*, 28(4):43–43, 2007.

[180] Earl D Sacerdoti. The nonlinear nature of plans. Technical report, STANFORD RESEARCH INST MENLO PARK CA, 1975.

[181] Avrim L Blum and Merrick L Furst. Fast planning through planning graph analysis. *Artificial intelligence*, 90(1-2):281–300, 1997.

[182] Henry Kautz and Bart Selman. Phusing the envelope: Planning, propositional logic, and stochastic search. (1-8):1194–1201, 1997.

[183] Jörg Hoffmann and Bernhard Nebel. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[184] Jörg Hoffmann. The metric-ff planning system: Translating"ignoring delete lists"to numeric state variables. *Journal of artificial intelligence research*, 20: 291–341, 2003.

[185] Alfonso Gerevini, Alessandro Saetti, and Ivan Serina. Planning through stochastic local search and temporal action graphs in lpg. *Journal of Artificial Intelligence Research*, 20:239–290, 2003.

[186] Alfonso Gerevini Alessandro Saetti Ivan Serina. An empirical analysis of some heuristic features for local search in lpg. 2004.

[187] Alfonso Gerevini, Alessandro Saetti, and Ivan Serina. Temporal planning with problems requiring concurrency through action graphs and local search. In *Twentieth International Conference on Automated Planning and Scheduling*, 2010.

[188] Manuel Sojer and Joachim Henkel. Code reuse in open source software development: Quantitative evidence, drivers, and impediments. *Journal of the Association for Information Systems*, 11(12):868–901, 2010.

[189] Martin Hitz and Behzad Montazeri. *Measuring coupling and cohesion in object-oriented systems.* na, 1995.

[190] Melissa JM Turcotte, Alexander D Kent, and Curtis Hash. Unified host and network data set. *ArXiv e-prints*, 2017.

[191] Derek Long, Maria Fox, and Richard Howey. Planning domains and plans: validation, verification and analysis. In *Proc. Workshop on V&V of Planning and Scheduling Systems*, 2009.

[192] Saddek Bensalem, Klaus Havelund, and Andrea Orlandini. Verification and validation meet planning and scheduling, 2014.

[193] Richard Howey, Derek Long, and Maria Fox. Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 294–301. IEEE, 2004.

[194] Lukáš Chrpa, Thomas Leo McCluskey, Mauro Vallati, and Tiago Vaquero. The fifth international competition on knowledge engineering for planning and scheduling: Summary and trends. *Ai Magazine*, 38(1):104–106, 2017.

[195] Mohammed Al Qady and Amr Kandil. Techniques for evaluating automated knowledge acquisition from contract documents. In *Construction Research Congress 2009: Building a Sustainable Future*, pages 1479–1488, 2009.

[196] Zhiyuan Chen, Arjun Mukherjee, and Bing 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)*, volume 1, pages 347–358, 2014.

[197] Felix Salfner, Steffen Tschirpke, and Miroslaw Malek. Comprehensive logfiles for autonomic systems. In *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, page 211. IEEE, 2004.

[198] Pinjia He, Jieming Zhu, Pengcheng Xu, Zibin Zheng, and Michael R Lyu. A directed acyclic graph approach to online log parsing. *arXiv preprint arXiv:1806.04356*, 2018.

Appendix A

Description of Microsoft events

According to Microsoft documentation[1], there are five categories of events that can be logged in Windows operating system. All event entries have structured data and an entry can only be of a single type. Table A.1 explains each category used by the event logging mechanism. Similar events are organised into separate collections that are explained in the following:

1. *Application* – all events generated by the applications installed on the machine;
2. *Security* – events triggered by the security auditing policies. All categories of security events are shown in Table A.2;
3. *Setup* – events related to the domain controller servers;
4. *System* – all events related to the Windows file system; and
5. *Forwarded* – events sent by other machines in the network.

[1] https://docs.microsoft.com/en-us/windows/desktop/eventlog/event-types

Event category	Description
Information	Describes about the successful operation performed by a user-application, hardware driver or background service. For example, if user starts a MS-SQL service on a machine, it will log an Information event 26022 with a message 'Server is listening on ['any' <ipv4> 1433]'.
Warning	Indicates an issue that is not significant at the moment, but might cause future problems. For example, if user attempts to restart the MS-SQL server and it fails, the event logging mechanism will log a Warning event 10010 with a message 'Application 'C:\Program Files\Microsoft SQL Server\130\LocalDB\Binn\sqlservr.exe' (pid 4984) cannot be restarted'. The warning events are still logged if the application or service recovers without losing data or operations.
Error	Notifies about a significant problem. For example, if the MS-SQL service fails to load upon user request, it will log an Error event 1000 with a message 'Faulting application name: sqlservr.exe ⋯'.
Success Audit	Describes the successful completion of an operation for which security auditing is enabled. For example, if a user tries to login and enters correct credentials, it will log a Success Audit event 4624 with a message 'An account was successfully logged on'.
Failure Audit	Records the failed attempt of an operation for which security auditing is enabled. For example, if a user tries logs on and enters incorrect credentials, it will log a Failure Audit success event 4625 with a message 'An account failed to log on'.

TABLE A.1: Event categories of a Microsoft Windows operating system

Category of security policy	Description
Account logon events	Stores all information regarding users logging on and off, timestamp, valid/invalid credentials, etc.
Account management	Keeps track of all activities related to user account, such as account creation and deletion, password change and so on.
Directory service access	Records all information regarding the system access control list.
Object access	Stores all events where user tried to access an object, such as printer, file, registry key, etc.
Policy change	Records all changes in security, audit and trust policies, and user right assignment.
Privilege use	Keeps track of all instances where user exercised authorised-permissions.
Process tracking	Stores activities of a process, such as duplication, start/exit, indirect resource access and so on.
System events	Stores all security events during the start and shutdown of a machine.

TABLE A.2: Security event categories of a Microsoft Windows operating system

Appendix B

Elements of PDDL-based representation

Every PDDL planning problem consists of two components: domain model and problem instance. The schemata are given in the following:

B.1 Domain model

```
(define (domain DOMAIN-NAME)
   (:requirements [:strips] [:equality] [:typing] [:adl] [:fluents])
   (:types T1 T2 ··· TN)
   (:predicates
      (PREDICATE-NAME-1 ?P1 ?P2 ··· ?PN)
      (PREDICATE-NAME-2 ?P1 ?P2 ··· ?PN)
      ...
   )
   (:functions
      (FUNCTION-NAME-1 ?F1 ?F2 ··· ?FN)
      (FUNCTION-NAME-2 ?F1 ?F2 ··· ?FN)
      ...
   )
   (:action ACTION-NAME-1
      :parameters (?P1 ?P2 ··· ?PN)
      :precondition (PRECOND-CONJUNCTION)
      :effect (EFFECT-CONJUNCTION)
   )
      ...
)
```

FIGURE B.1: Domain action model schema of PDDL

The domain schema is shown in Figure B.1, where elements written inside square brackets ('[]') depends on the user requirements. It contains the following elements:

1. *Domain-name* – This represents the domain's name, which allows alphanumeric characters, hyphens and underscores;

2. *Requirements* – As PDDL is a generic language, specifying syntax requirements makes it easier for the planning algorithm to interpret the domain model. For example, [:strips] shows that the current model uses STRIPS representation, [:fluents] depicts the usage of numeric fluents, etc.;

3. *Types* – These are used to represent the state variable types. The types are optional and require explicit definition by the user;

4. *Predicates* – These are the descriptive, logical expressions containing a set of parameters, which can be null as well. Each parameter is a variable name, starts with a question mark ('?') and can also have a type. The predicates have no intrinsic meaning in terms of domain definition;

5. *Functions* – These are the numeric fluents, which are used within actions to perform arithmetic operations; and

6. *Actions* – These represent the pre-condition and effect relationships among predicates to depict an operation. Each action has a name and takes a list of parameters along with their types. Both pre-conditions and effects contain the *and*/*or* conjunctions of one or more predicates, where each predicate can be formulated as either positive or negative. The negation is defined by a *not* operator. The effect conjunction can also have an action cost value, which is modelled and manipulated by numeric fluents.

B.2 Problem instance

```
(define (problem PROBLEM-NAME)
   (:domain DOMAIN-NAME)
   (:objects OBJECT-1 OBJECT-2 ··· OBJECT-N )
   (:init INITIAL-1 INITIAL-2 ··· INITIAL-N )
   (:goal GOAL-1 GOAL-2 ··· GOAL-N )
   (:metric MINIMIZE|MAXIMIZE (NUMERIC-FLUENT))
)
```

FIGURE B.2: Problem instance schema of PDDL

The problem schema is shown in Figure B.2, which contains the following elements:

1. *Problem-name* – This defines the problem instance name, which can have alphanumeric characters, hyphens and underscores;

2. *Domain-name* – This specifies the name of domain model, which will be used for this particular problem instance;

3. *Objects* – This is the list of constant objects, which are usually categorised into respective types. The type of an object is assigned using *Constant − ObjectType* syntax. The objects are used as arguments for the action parameters;

4. *Init* – This contains a list of predicates or facts that are true in the initial state. All other predicates are considered as false. The numeric fluents are also initialised here;

5. *Goal* – This contains a list of predicates, which are fully or partially satisfied upon successful planning process. The predicates with one or more parameters in both initial state and goal description are *grounded*, i.e. the variable parameters are replaced with constant objects; and

6. *Metric* – These are used to improve the quality of plan by maximising or minimising the accumulative value of numeric fluents encoded in domain actions.

Appendix C

Example of Temporal-Association-Causal (TAC) rules in HTML format

Figure C.1 presents an example of HTML representation of TAC rules that provides the following information to the user:

1. List of all TAC rules, where all events are depicted by their corresponding event types;
2. Complete information about the event type (by clicking the hyperlink). This information is collected from the online resources[1][2];
3. Temporality of all TAC rules;
4. List of one or more security event categories involved in each TAC rule; and
5. At the end, a summary stating number of total rules, rules with 100% temporality, rules belonging to a single category and single category rules with full temporality.

[1] https://docs.microsoft.com/en-us/windows/security/threat-protection/
[2] https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/

Following are the rules:

1: 4726 (A user account was deleted.) → 4729 (A member was removed from a security-enabled global group.) **Temporality: 1** and Category: User Account Management, Security Group Management

2: 4660 (An object was deleted.) → 4729 (A member was removed from a security-enabled global group.) **Temporality: 1** and Category: Other Object Access Events, Security Group Management

3: 4660 (An object was deleted.) → 5145 (A network share object was checked to see whether client can be granted desired access.) Temporality: 0.552631578947368 and Category: Other Object Access Events, Detailed File Share

4: 5145 (A network share object was checked to see whether client can be granted desired access.) → 5140 (A network share object was accessed.) Temporality: 0.7 and Category: Detailed File Share, File Share

5: 5140 (A network share object was accessed.) → 4705 (A user right was removed.) **Temporality: 1** and Category: File Share, Authorization Policy Change

6: 4705 (A user right was removed.) → 4700 (A scheduled task was enabled.) **Temporality: 1** and Category: Authorization Policy Change, Other Object Access Events

7: 5140 (A network share object was accessed.) → 4702 (A scheduled task was updated.) **Temporality: 1** and Category: File Share, Other Object Access Events

8: 5140 (A network share object was accessed.) → 4700 (A scheduled task was enabled.) **Temporality: 1** and Category: File Share, Other Object Access Events

9: 4705 (A user right was removed.) → 4702 (A scheduled task was updated.) Temporality: 0.75 and Category: Authorization Policy Change, Other Object Access Events

10: 4705 (A user right was removed.) → 4740 (A user account was locked out.) **Temporality: 1** and Category: Authorization Policy Change, User Account Management

11: 4720 (A user account was created.) → 4625 (An account failed to log on.) Temporality: 0.954545454545455 and Category: User Account Management, Logon

12: 4720 (A user account was created.) → 4728 (A member was added to a security-enabled global group.) **Temporality: 1** and Category: User Account Management, Security Group Management

13: 4732 (A member was added to a security-enabled local group.) → 4728 (A member was added to a security-enabled global group.) **Temporality: 1** and Category: Security Group Management

14: 4732 (A member was added to a security-enabled local group.) → 4738 (A user account was changed.) Temporality: 0.8125 and Category: Security Group Management, User Account Management

15: 4733 (A member was removed from a security-enabled local group.) → 4738 (A user account was changed.) Temporality: 0.8125 and Category: Security Group Management, User Account Management

16: 4740 (A user account was locked out.) → 4733 (A member was removed from a security-enabled local group.) Temporality: 0.5 and Category: User Account Management, Security Group Management

17: 4726 (A user account was deleted.) → 4740 (A user account was locked out.) **Temporality: 1** and Category: User Account Management

18: 5140 (A network share object was accessed.) → 4674 (An operation was attempted on a privileged object.) **Temporality: 1** and Category: File Share, Sensitive Privilege Use

19: 4705 (A user right was removed.) → 4985 (The state of a transaction has changed.) Temporality: 0.5384615384615385 and Category: Authorization Policy Change, File System

20: 4740 (A user account was locked out.) → 4724 (An attempt was made to reset an account's password.) **Temporality: 1** and Category: User Account Management

21: 4724 (An attempt was made to reset an account's password.) → 4732 (A member was added to a security-enabled local group.) **Temporality: 1** and Category: User Account Management, Security Group Management

22: 4732 (A member was added to a security-enabled local group.) → 4720 (A user account was created.) **Temporality: 1** and Category: Security Group Management, User Account Management

23: 4724 (An attempt was made to reset an account's password.) → 4728 (A member was added to a security-enabled global group.) **Temporality: 1** and Category: User Account Management, Security Group Management

24: 4660 (An object was deleted.) → 4726 (A user account was deleted.) **Temporality: 1** and Category: Other Object Access Events, User Account Management

25: 4729 (A member was removed from a security-enabled global group.) → 5145 (A network share object was checked to see whether client can be granted desired access.) Temporality: 0.552631578947368 and Category: Security Group Management, Detailed File Share

26: 4729 (A member was removed from a security-enabled global group.) → 5140 (A network share object was accessed.) Temporality: 0.8 and Category: Security Group Management, File Share

27: 4660 (An object was deleted.) → 4705 (A user right was removed.) **Temporality: 1** and Category: Other Object Access Events, Authorization Policy Change

28: 5140 (A network share object was accessed.) → 4738 (A user account was changed.) **Temporality: 1** and Category: File Share, User Account Management

29: 4722 (A user account was enabled.) → 4720 (A user account was created.) **Temporality: 1** and Category: User Account Management

30: 4722 (A user account was enabled.) → 4728 (A member was added to a security-enabled global group.) **Temporality: 1** and Category: User Account Management, Security Group Management

31: 4724 (An attempt was made to reset an account's password.) → 4738 (A user account was changed.) Temporality: 0.5 and Category: User Account Management

32: 4733 (A member was removed from a security-enabled local group.) → 4724 (An attempt was made to reset an account's password.) **Temporality: 1** and Category: Security Group Management, User Account Management

33: 4985 (The state of a transaction has changed.) → 4733 (A member was removed from a security-enabled local group.) Temporality: 0.5769230769923077 and Category: File System, Security Group Management

34: 5145 (A network share object was checked to see whether client can be granted desired access.) → 4705 (A user right was removed.) **Temporality: 1** and Category: Detailed File Share, Authorization Policy Change

35: 5140 (A network share object was accessed.) → 4625 (An account failed to log on.) Temporality: 0.7272727272727727 and Category: File Share, Logon

36: 5140 (A network share object was accessed.) → 4733 (A member was removed from a security-enabled local group.) Temporality: 0.6 and Category: File Share, Security Group Management

37: 4724 (An attempt was made to reset an account's password.) → 4722 (A user account was enabled.) **Temporality: 1** and Category: User Account Management

38: 4705 (A user right was removed.) → 4733 (A member was removed from a security-enabled local group.) Temporality: 0.5 and Category: Authorization Policy Change, Security Group Management

39: 4702 (A scheduled task was updated.) → 4733 (A member was removed from a security-enabled local group.) Temporality: 0.5 and Category: Other Object Access Events, Security Group Management

40: 4700 (A scheduled task was enabled.) → 4733 (A member was removed from a security-enabled local group.) Temporality: 0.5 and Category: Other Object Access Events, Security Group Management

41: 4740 (A user account was locked out.) → 4722 (A user account was enabled.) **Temporality: 1** and Category: User Account Management

42: 5140 (A network share object was accessed.) → 4985 (The state of a transaction has changed.) Temporality: 0.8769230769923077 and Category: File Share, File System

43: 5145 (A network share object was checked to see whether client can be granted desired access.) → 4985 (The state of a transaction has changed.) Temporality: 0.9149797570850202 and Category: Detailed File Share, File System

Description	Value
Number of rules	43
Number of rules with full temporality	24
Number of rules single category	7
Number of single category rules with full temporality	6

FIGURE C.1: An example of TAC rules represented in the HTML format.