Learning Domain Knowledge for Automated Security Administration

Saad Khan and Simon Parkinson
1 Department of Computer Science, University of Huddersfield
Huddersfield HD1 3DH, UK
{saad.khan, s.parkinson}@hud.ac.uk

Abstract

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

Introduction

Many organisations are vulnerable to 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 (Viduto et al. 2012). Businesses are facing challenges with recruiting and maintaining expertise within their organisation and as a result, they are often unable to adequately secure their systems.

The security events provide knowledge that describes both unauthorised activities and configuration events. Most of the current solutions use event logs for identifying problems; however, there is a great potential to adopt this same philosophy for identifying security issues. It would be of great benefit to automatically determine and extract pattern of events that detail administrative activities performed on a machine. As a simplistic 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. Despite being simple, this example describes the fact that managing security requires expert knowledge. This action-based knowledge has value for automating security actions. However, a fundamental limitation exists in that someone to manually extract and model the administrative actions.

The question addressed in this paper is whether it is possible to automatically extract and effectively utilise security administrative actions (i.e., domain knowledge) from the event log entries of a machine. Exploring this question lead to the development of an unsupervised knowledge acquisition technique that can discover relationships among the events and use to determine a course of action for security improvements on a previously unseen computer. The developed technique is tested on Microsoft Windows OS (Simache, Kaâniche, and Saidane 2002), but it is independent of OS and can easily be applied to the event logs from other sources. Following are steps that have been developed to perform automated knowledge acquisition:

1. Convert event log entries to an object-based model;
2. Determine both minimum and maximum support values from the object-based model;
3. Identify object-based rules from the model using the support values in an association rule mining algorithm;
4. Determine the event-based rules, such that any two events are associated by the object-based rules;
5. Improve the quality of relationships based on the temporal order of events;

*This paper has been already published.
Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
6. Combine and expand the individual relationships to form
and validate sequences of events;
7. Convert the sequences of events into directed acyclic
graph and infer a causal rank for each relationship; and
8. Encode all relationships along with all relevant informa-
tion into a PDDL domain action model.

The paper is organised as follows: First, a brief review of
existing domain learning techniques is provided. Next, the
process of automated knowledge acquisition is presented.
This includes the process of combining individual rules to
form sequences of events, where each sequence represents a
single configuration activity. The next part presents the pro-
cess of assigning a causal rank score to each rule based on
the level or strength of causality. Following on, a method
to encode the extracted relationships into an action model
is presented, which leads on to the explanation of the au-
tomated method for extracting the problem instance from
vulnerable machine, and applying AP technique to generate
plan. Finally, an empirical analysis of the developed solution
on the event logs of 21 live systems.

Related work

Research into domain knowledge acquisition can loosely be
classed into manual an automated approached. For manual
domain model construction, there are several tools available
that provide interface for the design, validation and verifi-
cation of domain models. For example, itsSIMPLE (Vaquero
et al. 2012) uses Object Oriented architecture and Unified
Modelling Language (UML) to enable the users for building,
visualising and modifying domain models. GIPO (Graphical
Interface for Planning with Objects) (Simpson et al. 2014),
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 sev-
eral 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) bar-
reiro2012europa, was developed by NASA to tackle real-
world problems.

Researchers have also produced several automated do-
main learning and modelling techniques. The Opmaker (Mc-
Cluskey et al. 2009) algorithm was created to output a com-
plete domain action model using a partial domain model
and a set of training data for planning. This tool automati-
cally determines the intermediate transition states by track-
ing and generalising the object changes provided in the train-
ing set. A tool called LOCM (Learning object-centric mod-
els) (Cresswell, McCluskey, and West 2013) was developed
that induces the domain model by allowing multiple state
machines to represent a single object using Finite State Ma-
achines characterised by a set of transitions. A major limita-
tion of LOCM is lack of compatibility with domains hav-
ing static facts; however, this issue has been resolved in
Automated Static Constraint Learner or ASCol (Jilani et
al. 2015). The ASCol exploits a directed graph representa-
tion of operator arguments in a plan solution and uses
that to discover pairwise static relations among precondi-
tions and effects. Another improvement of LOCM is pre-
semed by LC_M algorithm (Gregory, Lindsay, and Porteous
2015). This algorithm is capable of filling missing data using
plan traces as well as identify structural elements that might
constitute as noise. A recent paper presents another domain
learning tool, called SRMLearn (Arora et al. 2017). This
tool represents inter and intra-action relationships as a max-
imum satisfiability problem (MAX-SAT) and solves it with
a weighted MAX-SAT solver to produce the domain model.
Other tools also exit that employs Artificial Intelligence (AI)
techniques, such as Natural Language Processing (Lindsay
et al. 2017) and Long Short-Term Memory (Arora et al.
2018), to construct automated domain models.

Despite the advantages of state-of-the-art domain learning
solutions, a major drawback is the need for pre-existing do-
main knowledge in the form of plan traces. Most of existing
solutions require plan traces in some form, and as such con-
ducting autonomous learning in a previously unseen applica-
aproblem area would first require the construction of a plan trace. In
Many real-world 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 guaran-
teed to be always available. These issues can be resolved
by applying different techniques, like, crowd-sourcing to ac-
quire action models (Zhuo 2015) or learning from human
demonstrations to perform case-based planning (Long et al.
2009). However, such techniques are expensive and require
large amount effort to prevent human-errors and perform
conflict resolution. This motivates the research challenge of
developing an autonomous approach to acquire and use se-
curity 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.

Automated knowledge acquisition

Considering the importance of creating an automated do-
main model, our research pursues potential data correlation
 techniques that are unsupervised and can discover relations-
ships among a diverse set of event log entries. The proposed
solution employs the Association Rule Mining (ARM) tech-
nique, which relies on support and other metrics to deter-
mine the correlations among data items. It enables searching
for valuable information based on frequency and is combi-
natorial in nature. The remaining section explains the auto-
mated mechanism of acquiring knowledge from Microsoft
security event logs that contains the expert knowledge.

Preparing Object-Based Model

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, permis-
sion levels, security identification etc. In the following discussion, \( D \) is used to model a dataset of event entries, where \( D = \{ E_1, E_2, \ldots, E_N \} \). The event, \( E = (id, O) \), where \( id \) is a numeric event type, and \( O = \{ o_1, o_2, \ldots, o_n \} \) is the set of event objects. The set \( O \) belongs to set of unique objects, \( I = \{ o_1, o_2, \ldots, o_n \} \).

**Association Rule Mining**

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 utilized to discover correlation between events.

**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 frequently co-occurring, the respective events are to co-occur too. The proposed solution uses Apriori algorithm (Agrawal, Srikant, and others 1994) for ARM. The association rules are discovered within a tabular dataset of objects that uses different measures to mine interesting results. The reason behind using Apriori algorithm is that it performs exhaustive search in the dataset to produce large number of frequent itemsets (Garg and Kumar 2013), 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. Researchers have demonstrated that the ARM technique generates a high quantity of rules and has a complexity of \( O(n^2) \), where \( n = |I| \) (Sarma and Mahanta 2012).

**Establishing Support Value:** ARM takes as input both minimum (\( \text{minsup} \)) and maximum (\( \text{maxsup} \)) support values and finds all those rules having \( \text{minsup} \leq \text{support} \leq \text{maxsup} \). 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 within the limits of SR (Vamanala, Sree, and Bhavani 2014). Choosing the right support for ARM algorithm is critical for generating rules of better quality and quantity. There is a need to define both minimum and maximum support values, 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 challenging.

In the presented work, SR is calculated based on the object frequency distribution (OFD) of events types. 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. The distinction between a normal and abnormal distribution is performed using a normality test. Many methods are available to determine whether a given distribution is normal. We have used Two-Sample Kolmogorov–Smirnov (TSKS) test as it is suitable for large sample sizes (Xiao 2017). 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.

**Event-Type Association Rules:** 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 (\( \text{matched} \)), the number of objects missing from a rule but exists in the event entry (\( \text{missing} \)) and number of objects that are in the rule but missing from event entry (\( \text{additional} \)). The formula used to determine similarity is \( \frac{\text{matched}}{(\text{matched} + \text{missing} + \text{additional})} \). 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.

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.

**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.

**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. This research presents 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 \( (x_1, x_2) \rightarrow (y_1, y_2) \), 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
Algorithm 1 Creating temporal-association rules.

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

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

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

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. 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. 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.

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. The threshold value is flexible and can range from $50\%-100\%$. We selected this threshold value to create a balance between the quality and quantity of rules by tolerating somewhat ‘inaccurate’ rules, rather than having none at all.

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 subset combinations 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.

The next step is to create an ordered set of events, 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 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.

At this stage, temporally associated events have been arranged into appropriate sequences, which can describe the set of actions required to perform a certain administrative task; however, correlation does not necessarily imply causality. 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.

**Determining Causality**

Causality defines what (one or more) activities in the underlying machine led to a particular activity. 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) (Spirtes and Glymour 1991) and the subsequent improved version named Fast Causal Inference (FCI) (Spirtes, Glymour, and Scheines 2000). For this research, the FCI algorithm is applied as it allows hidden variables, is scalable to high-dimensional data and has better accuracy. Note that a detailed explanation of FCI algorithm is beyond the scope of this paper.

The FCI algorithm takes a directed acyclic graph (DAG) as an input. A DAG is a type of graph where every edge is directed, i.e. does not have conflicting edges and cycles. To prepare the input, the first step is to iterate over all temporally associated sequences and create subset (one-to-one event) rules of relationships. The next step is to remove conflicting and cyclic subset rules based on temporal-association accuracy (TAA) value and generate a DAG. The FCI algorithm outputs a Partial Ancestral Graph (PAG), which has been proven (Zhang 2008b) to be maximally informative as it is capable of representing all types of edges: directed, undirected, partially directed and double-headed. Following is interpretation of output PAG (Zhang 2008a) with respect to proposed solution’s context. This conversion of temporal-association into causal rules, referred as temporal-association-causal rules, provides an additional layer of confidence in terms of reliability and accuracy. 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 with the highest rank 4;
2. \( X \circ \rightarrow Z \): the algorithm is certain that \( Z \) is not the cause of \( X \), but not sure about the other way around. We have assigned this a rank 3;
3. \( X \circ \circ \rightarrow Z \): the algorithm is uncertain about whether \( X \) caused \( Z \) or \( Z \) caused \( X \). This uncertainty occurred because both of these relationships were found. 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 \( X \). We have assigned this a rank 1 due to the lowest 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.

**Knowledge Representation and Utilisation**

Utilising the generated temporal-association-causal (TAC) rules on a previously unseen machine is not feasible as algorithmic support is required. As the problem of selecting rules to use is a deliberation problem, we consider the use of an automated deliberation techniques in the form of AP (Ghallab, Nau, and Traverso 2004). The TAC rules present a collective set of expert actions that can be applied to any machine. However, it is unknown which actions will serve the purpose, i.e., only a subset of actions might be applicable and useful. Furthermore, there is no certainty that the order of acquired actions is same as the order that would be applied on an unseen system. Therefore, a clear decision-making process is needed regarding which actions to select for any given machine. This deliberation is traditionally performed by a human expert, which requires a significant time and effort. In this work, we consider the use of AP to replace manual effort. Hence the final stage of the research is to model the TAC rules into discrete actions and utilise AP techniques to increase the usability of the proposed solution.

The proposed solution uses Planning Domain Definition Language (PDDL) to represent TAC rules in a common format for representation to allow better knowledge distribution and more direct comparison of systems and approaches (Fox and Long 2003). Having PDDL as a common format for representation allows better knowledge distribution and more direct comparison of systems and approaches. The PDDL representation of extracted rules enables the proposed solution to utilise AP algorithms to produce a plan of actions.

**Domain Modelling**

In a PDDL domain action, the cause event of the TAC rule becomes the precondition. The name of an action is selected as ‘LHS event-to-RHS event’. The combined list of object names from both cause and effect of a rule constitutes as parameters. Each domain action having one or more parameters will provide all objects that are required to make any administrative change in the security of underlying system. It should be noticed here that the domain actions are same as the TAC rules.

A predicate is placed in each action’s effect to avoid the successive selection of the same actions, which involve the same objects. It is false before the action execution and means that the given objects have not been pro-
Set of actions or those with the highest accumulative-weight value in the available planning time, the maximisation metric is applied in the problem file. The PDDL does not allow multiple metrics in a single instance, hence the reason for using the product of counter, TAA and causal rank values as accumulative-weight. The complete set of actions provided against the identified issues based on the knowledge encoded in domain model ensure to aid the non-experts to identify the security risks. The total accumulative-weight value of a plan is incremented by the accumulative-weight of each chosen action. A live system problem file is shown in Figure 2, which was extracted from a machine having poor security. The objects and current security state in the form of event types are extracted from the log entries of the machine. For example, Test1 - DisplayName represents an object Test1, which is of DisplayName type. Same goes for the WORKGROUP - SubjectDomainName, where domain name of underlying subject is WORKGROUP. The accumulative-weight value is initialised with zero and will be incremented (to maximum extent) according to the execution of domain actions.

Plan generation

The Local search for Planning Graphs (LPG) (Gerevini, Saetti, and Serina 2003) planner is used due to its good performance and support for PDDL. The planner was executed in incremental mode to identify plans of increasing quality within a specified time. The order of plan actions is crucial to successfully carry out the security-related administrative tasks. For sake of an example, a small segment of plan extracted from a live system with limited set of objects is shown in Figure 3. The three actions describe the process of creating a secure account in a networked environment. The objects of each action describe the information needed to perform the action. The object names are given in Figure 2. The event type 4720 shows that a user account IE8SAADS was created in a network by another account Administrators. After that, Administrators...
Figure 3: A small segment of plan for the target machine using the automated domain knowledge acquisition.

added the IE8SAADS account to local and global security groups as evident by event 4732 and 4728, respectively. At the end, event type 4722 shows that the account IE8SAADS was enabled. Although it is a small segment of a plan, these actions inform the non-experts to assign appropriate user-group when creating a new user account for better security.

Results and discussion

Following steps are performed in this the evaluation:
1. Acquire event log data from 21 machines;
2. Use the developed solution to produce domain models;
3. Produce a set of manual domain action models using the same data sources (created by human experts);
4. Create a problem instance for each vulnerable machine;
5. Generate automated and manual plan traces individually using an automated planning algorithm; and

The empirical analysis presented in this paper was performed on a 32-bit Microsoft Windows 7 virtual machine with 2 cores of Intel i7 3.50GHz CPU and 4GB RAM. Table 1 presents the results from each layer of processing, such as number of event log entries, rules, plan, execution time, accuracy etc. It should be noticed that the table only mentions TAC rules but not the domain actions as both are the same. The acquired domain models from 21 datasets have presented suitable accuracy, ranging from 73% – 92%. The solution has also shown reasonable performance, taking 39 minutes for processing 69,854 entries and 31 minutes for processing 313,470 entries. It has also been observed that datasets with comparatively lower accuracy contain such rules, where one event is linked to many others. Here AP plays an important role to determine the suitable set of actions with respect to accumulative-weight values.

Comparison of manual and automated plans: This segment presents an example of comparison between an automated and manual plan, and demonstrates the ability of the proposed solution to suggest human-like actions. Both complete plans, including security identification and mitigation actions, are shown in Table 2. We have also created a parsing tool that uses event log entries of the vulnerable machine and the domain action model to describe the security issue and respective solution, along with elaborating each action of the plan. The accuracy calculation mechanism covers all aspects by finding the number of correct, incorrect, additional and missing actions from any automated plan by comparing it to the corresponding manually generated plan.

Interpretation of a plan: A timeline demonstrating the complete security concern and mitigation (from Table 2) is presented in Figure 4. The red and blue coloured boxes show the events found in vulnerable machine, whilst the green coloured box shows the mitigation plan. The timeline consists of the full list of actions included in the plan; however, it is important to communicate that a large portion of the actions have already been discovered in the event log of a vulnerable machine. This section also demonstrates that the proposed solution is easy to use and an automated plan only requires a basic knowledge for interpretation. Following is the brief interpretation of planner output:

- **Matched** – This part of the plan shows the security issue of the vulnerable machine and was identified by matching one or more initial state(s) of the problem instance with the domain action model:

  - **User Logon** – Event 5145 shows that a certain user, with a $S_{1.5.18}$ group security identifier (SID) and IEUser1 username, tried to access file-share service on a remote server. The group SID is a unique value of variable length used to identify a trustee and issued by an authority. The event 5140 shows that server access was allowed for IEUser1. After that, the remote server requested the login credentials from IEUser1. The required username and password were provided to the server as evident by event 4648; and

  - **Privilege Escalation attack** – Following the login, event 4657 is triggered which shows that the IEUser1 added a new registry value. After that, event 4798 informs that the IEUser1 gathered local users of the server. The next two events (4647 and 4624) reveal that the IEUser1 logged out of the server, and then successfully logged back in, but, with a different group SID $S_{1.5.19}$.

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

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

Based on the sequence of events, it shows that an administrator found a suspicious activity in terms of system registry modification by a local user account, which lead to the discovery of possible privilege escalation attack on a remote server, running file-share service. Initially it is unknown whether the account became an actual member of a different user-group, but it became clear on a second login when the security identifier (SID) changed from $S_{1.5.18}$ to $S_{1.5.19}$. The registry alteration most probably allowed the user to switch user-group that has higher access permission and can compromise the service. The plan also shows...
Table 1: Results of empirical analysis from domain learning and automated planning.

<table>
<thead>
<tr>
<th>Event logs</th>
<th>Rule Mining and Domain Modelling</th>
<th>Automated Planning</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of events</td>
<td>Minimum Support</td>
<td>Maximum Support</td>
</tr>
<tr>
<td>1</td>
<td>5,027</td>
<td>0.07</td>
</tr>
<tr>
<td>2</td>
<td>8,253</td>
<td>0.08</td>
</tr>
<tr>
<td>3</td>
<td>521</td>
<td>0.25</td>
</tr>
<tr>
<td>4</td>
<td>1,122</td>
<td>0.05</td>
</tr>
<tr>
<td>5</td>
<td>1,079</td>
<td>0.05</td>
</tr>
<tr>
<td>6</td>
<td>1,691</td>
<td>0.05</td>
</tr>
<tr>
<td>7</td>
<td>1,714</td>
<td>0.13</td>
</tr>
<tr>
<td>8</td>
<td>9,721</td>
<td>0.08</td>
</tr>
<tr>
<td>9</td>
<td>9,770</td>
<td>0.05</td>
</tr>
<tr>
<td>10</td>
<td>13,026</td>
<td>0.06</td>
</tr>
<tr>
<td>11</td>
<td>10,148</td>
<td>0.05</td>
</tr>
<tr>
<td>12</td>
<td>8,832</td>
<td>0.07</td>
</tr>
<tr>
<td>13</td>
<td>3,403</td>
<td>0.05</td>
</tr>
<tr>
<td>14</td>
<td>31,719</td>
<td>0.11</td>
</tr>
<tr>
<td>15</td>
<td>1,072</td>
<td>0.07</td>
</tr>
<tr>
<td>16</td>
<td>1,991</td>
<td>0.05</td>
</tr>
<tr>
<td>17</td>
<td>826</td>
<td>0.05</td>
</tr>
<tr>
<td>18</td>
<td>5,309</td>
<td>0.06</td>
</tr>
<tr>
<td>19</td>
<td>69,854</td>
<td>0.05</td>
</tr>
<tr>
<td>20</td>
<td>313,470</td>
<td>0.05</td>
</tr>
<tr>
<td>21</td>
<td>32,688</td>
<td>0.15</td>
</tr>
</tbody>
</table>

Table 2: Comparison of automated and manual plans alongside relevant objects.

<table>
<thead>
<tr>
<th>Automated plan</th>
<th>Manual plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actions</td>
<td>Relevant Objects</td>
</tr>
<tr>
<td>1. 5145 → 5140</td>
<td>Share name: Users, Subject: IEUser1, SID: S_1_5_18</td>
</tr>
<tr>
<td>2. 5140 → 4648</td>
<td>Target-domain: IE8Win7, server: localhost</td>
</tr>
<tr>
<td>3. 4647 → 4579</td>
<td>Operation: registry value added, Type: Multi-String</td>
</tr>
<tr>
<td>4. 4579 → 4798</td>
<td>Subject: IEUser1, Domain: WORKGROUP, Process: mmc</td>
</tr>
<tr>
<td>5. 4798 → 4647</td>
<td>Subject: IEUser1</td>
</tr>
<tr>
<td>6. 4647 → 4024</td>
<td>Subject: IEUser1, SID: S_1_5_19, Domain: WORKGROUP</td>
</tr>
<tr>
<td>7. 4024 → 4726</td>
<td>Subject: IEUser1, Domain: WORKGROUP</td>
</tr>
<tr>
<td>8. 4726 → 4729</td>
<td>Target: IEUser1, SID: S_1_5_19, Subject: Administrator</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 4: A complete timeline of events extracted in the form of a plan.
the registry location where a new string value was added: HKLM\Software\Microsoft\Windows\currentversion\run. The administrator resolved this issue by removing the user-account from the server.

**Conclusion**

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

**References**


Gregory, P.; Lindsay, A.; and Porteous, J. 2017. Domain model acquisition with missing information and noisy data. In *Workshop on Knowledge Engineering for Planning and Scheduling (KEPS). The 27th International Conference on Automated Planning and Scheduling (ICAPS)*.


