Causal Connections Mining Within Security Event Logs

Saad Khan and Simon Parkinson
University of Huddersfield
Queensgate, Huddersfield, HD1 3DH, UK
⟨firstname.lastname⟩@hud.ac.uk

ABSTRACT

Performing both security vulnerability assessment and configuration processes are heavily reliant on expert knowledge. This requirement often results in many systems being left insecure due to a lack of analysis expertise and access to specialist resources. It has long been known that a system's event log provides historical information depicting potential security threats, as well as recording configuration activities. In this paper, a novel technique is developed that can process security event logs on a computer that has been assessed and configured by a security professional, and autonomously establish causality amongst event log entries to learn performed configuration tasks. This extracted knowledge can then be exploited by non-professionals to plan steps that can improve the security of a previously unseen system.

CCS CONCEPTS

•**Information systems** → *Association rules;* •**Computing methodologies** → *Causal reasoning and diagnostics;*

KEYWORDS

Knowledge extraction, Automated, Association, Causal

1 INTRODUCTION

Most organisations are facing security threats exposed by their digital infrastructure, and given the increasing size and critical nature of their business operations, there is a need to pro-actively identify and mitigate security vulnerabilities. It requires expert knowledge of the latest security threats and how they can be mitigated. Such knowledge is in short supply but is needed to keep the system secure. This paper presents a solution that can extract security actions performed on a system using only event log data. An event log contains semi-structured information regarding the activities within the system, generated by either a user, application or the system itself [13]. Event logs are beneficial for tracking (step-by-step) state changes in the system. This paper focuses on finding causal connections among security event log entries of Microsoft Windows operating systems (OS) [14]. Identifying a cause and effect relationship amongst two or more security event log entries can help in determining *what* problem was identified within the system, *how* it was identified, and *what* remedial action was performed.

Security event logs possess information on pre-defined auditable security and configuration policies, and record events such as login and logout attempts, permissions change, policy change, and so on. According to Microsoft documentation[1], each entry contains a unique identifier (ID), event source, user account, time-stamp, machine name, and a brief description, which provides specific information depending on the event type, such as failure reason, error codes etc. These properties provide knowledge that describes what was changed in the system.

The aim of this research is to automatically identify related security actions performed in the system and allow non-experts to utilise the knowledge to audit their systems. This approach introduces an unsupervised learning technique for identifying causal relationships amongst event log entries. The entries are modelled based on their information and are used as an input for correlation mining process. The generated rules can assist in identifying causal connections amongst events, unsupervised and without programmatically encoding knowledge. The performance and accuracy of the proposed solution demonstrates suitability.

The remainder of the paper is organised as follows: the first section surveys ARM and causal relationships techniques; the second section presents how to determine the causal connections among security event logs entries; and finally, the last section includes empirical analysis of solution using live computers.

2 RELATED WORK

Association Rule Mining (ARM) [2] is a method of pattern analysis that is commonly used to reveal interesting relationships among seemingly unrelated elements. The ARM process also includes two key concepts, support and confidence, to determine the most relevant relationships [3]. The support determines the frequency of an item appearing in the data, while confidence is the number of times a rule is found to be true. A commonly used ARM algorithm, called Apriori, was developed by [1]. It uses support value based pruning and prevents exponential growth of possible frequent itemsets. Many algorithms are available that have improved the efficiency of itemset generation in Apriori [16, 17].

It should be noted here that the presence of association rules do not necessarily imply causality. Many studies propose mechanisms to convert correlation into causal relationships. A recent study suggests that casual relationship can be defined by finding a measure of interestingness for association rules using strong feature selection data [5]. The interestingness measure is a way of checking the relation between items of an extracted rule and can be measured using an exclusive causal-leverage mechanism [4]. Another study presents the Causal Association Rule discovery (CAR) algorithm, which integrates ARM technique with the cohort study to determine

K-CAP 2017, December 4–6, 2017, Austin, TX, USA
© 2017 ACM. ISBN 978-1-4503-5553-7/17/12…$15.00
DOI: https://doi.org/10.1145/3148011.3154476

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

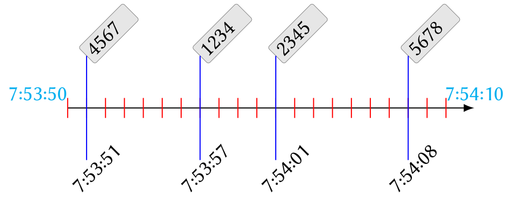

Figure 1: The visual time-line representation of D covering a time span of 20 seconds.

potential causal rules in observational data [7]. Other research work has been conducted to determine the cause behind certain system failures [11], where correlation mining is performed on a limited set of event log entries and uses time-series methods and rule-based classification to create causal relationships. These approaches do not provide a suitable solution for identifying time-based causal connections in data. They either rely on manual input or works in a constrained and predictable environment. Additionally, due to lack of considering temporal aspects, the existing techniques are unable to define the origin of sequence of causal events.

3 MINING EVENT LOG RELATIONSHIPS

3.1 Preparing Event Logs

In the following discussion D is used to model the set of event entries, where $D = \{E_1, E_2, ..., E_N\}$. Here E is a double tuple $E = \{ID, O\}$. ID is a numeric event ID, and O is the set of unique objects, such that $O = \{O_1, O_2, ..., O_N\}$. From the on-going example, $D = \{E_1, E_2, E_3, E_4\}$ where:

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

Initial processing is necessary to ensure that any whitespace and special characters are removed to maintain efficiency. In this trivial example, each entry has 4 objects along with their event IDs. The objects of an entry are separated by a comma and represent the information such as username, OS, applications and network ports. Each entry starts with the corresponding event ID. Events occur over the duration that the system is running and they often appear in sequence, although concurrency can also occur. An sequential example is presented in Figure 1 where the events occur over a 20 second period. Each entry contains a timestamp denoting the time that the event log entry was created. The example provided in Figure 1 is a synthetic example used throughout the paper to help communicate the technical approach developed in this paper.

3.2 Object-based Rules

This section presents a technique to determine correlation rules among event logs entries using ARM processes. The purpose is to identify events that are likely to occur together using their objects. ARM is utilised as a method for describing, analysing and presenting association rules (ARs) extracted from tabular databases [9] that satisfy the condition of co-occurrence. ARM consists of two fundamental steps that are taken to extract interesting ARs [6]: (i) frequent itemset – find the set of items that appears many times in data and (ii) correlation – identify strong co-occurrences amongst the frequent itemset. The interesting factor is discovered using support (frequency of itemset) and confidence (correctness of AR) values [3]. In the following discussion, D is the set of event log entries E and $I = \{O_1, O_2, ..., O_n\}$ contains the total set of unique objects from all event entries.

Researchers have also demonstrated that ARM generates a large quantity of rules and has a time complexity of $O(N^2)$ where N is the total number of unique elements across all log entries [12]. Every ARM algorithm takes minimum support (*minsup*) and confidence ($minconf$) to find all those rules having support $\geqslant$ *minsup* and confidence $\geqslant minconf$. Although any rule with a greater support and confidence values is relevant [15], but they still might not be interesting to the users.

3.3 Event-based Rules

At this stage, relationships amongst an event's objects have been identified. However, it is then necessary to translate these relationships to learn connections amongst complete event entries. This is done through a process of searching and counting the frequency of object-based relationships amongst two events.

Object-based rules are in the form of $X \implies Y$, where X and Y contains at least one object each. Converting them into event-based rules requires searching the objects from both X and Y individually within all event log entries. If one or more matches are found between the objects of X and Y and all event log entries, their respective event IDs are extracted and placed in the same order and manner as of original object-based rules, hence forming a new set of event-based rules ($r \in R$). The process also requires a condition that the event IDs on LHS and RHS of implication must not be equal as it would create unusable cyclic redundancy. In the continuing example, consider the object-based rule $\{User1\} \implies \{Win7\}$. The $User1$ is found in the log entries with event ID as 1234 and 4567, whilst $Win7$ is found in 4567 and 5678. As 4567 is already found in LHS, the event-based rule would be $\{1234, 4567\} \implies \{5678\}$.

3.4 Cause and Effect Relationships

The technique to determine the *cause* and *effect* relationships among events is presented in Algorithm 1. It is based on time as any effect should not occur before its cause events. It starts by iterating over each event-based rule and split the composite (containing one or more event IDs) LHS and RHS into individual 'sub-rules'. For example, if we consider the rule (1234,4567 $\implies$ 5678), the sub-rules would be; (1234 $\implies$ 5678) and (4567 $\implies$ 5678).

The next step is to determine the truthfulness of all sub-rules from each event-based rule to facilitate the conversion of correlation rules into causal connections. The truthfulness value is a temporal metric, which defines the accuracy of causal relationship as stated by Hill's criteria [10]. It involves finding all indices of event IDs, where they occurred in database D, from both LHS and RHS of every sub-rule. Assume there are 200 event log entries instead of 4 in the previous example. Considering the sub-rule (1234 $\implies$ 5678), the event ID from LHS (1234) might potentially occur at $E_9, E_{36}, E_{59}, E_{73}, E_{105}$ and the event ID from RHS (5678) at

Algorithm 1 Determining cause and effect relationships from associative rules.

```
Input: Set of event-based rules $R = \{r_1,...,r_n\}$, where $r = (r_x \implies r_y)$, and $r_x$ and $r_y$ consist of at-least one $EventID$ each
Input: Set of ordered event logs entries $D$ containing 2-tuple of $EventIDs$ and their corresponding objects
Output: Set of cause and effect rules $C = \{(c_1, tr_1),...,(c_n, tr_n)\}$, where $c = (c_x \implies c_y)$ and $c_x$ is cause, $c_y$ is effect and $tr$ is the truthfulness value
 1: procedure Cause-Effect-Relationship
 2:     $C = \emptyset$
 3:     for all $r_i \in R$ do
 4:         $r_x, r_y = r_i$
 5:         $tct = 0$
 6:         for all $EventIDs_x, EventIDs_y \in r_x, r_y$ do
 7:             $ct = 0$
 8:             $PosE_x$ = GetIndicies($EventIDs_x, D$)
 9:             $PosE_y$ = GetIndicies($EventIDs_y, D$)
10:             for all ($Pos_{x_k} \in PosE_x$) and ($Pos_{y_k} \in PosE_y$) do
11:                 if $Pos_{x_k} < Pos_{y_k}$ then
12:                     $ct$ += 1
13:                 end if
14:             end for
15:             $tct$ += $ct$/sizeof($PosE_x$)
16:         end for
17:         $tr_i = tct$/sizeof$(r_i) \times 100$
18:         if $tr_i \geq 50$ then
19:             $C.Add(r_i)$
20:         end if
21:     end for
22: end procedure
```

$E_{21}, E_{43}, E_{57}, E_{88}, E_{112}$. After both lists of indices, $PosE_x$ and $PosE_y$, are identified on line 8 and 9, respectively. Their elements are compared using a loop on line 10. Every time an index from $PosE_x$ is found to be less than the index from $PosE_y$ on line 11, a counter, *ct*, is incremented on line 12. After the comparison is performed, *ct* is divided by the total number of elements in $PosE_x$ to find a ratio of how many times LHS event occurred before RHS (line 15).

In the continuing example, ratio of (1234 $\implies$ 5678) would be 4/5 = 0.8. The same steps are repeated for remaining sub-rules using an iterator on line 6. After calculating ratios of every sub-rule in an event-based rule, their combined validity percentage is determined on 17. For example, assume the ratio of sub-rule 4567 $\implies$ 5678 as 0.89. The overall percentage of (1234,4567 $\implies$ 5678) would be $(0.80 + 0.89)/2 \times 100 = 84.5\%$. The algorithm processes all event-based rules in this way and all those rules having overall certainty level of 50% or above are chosen for further processing.

3.5 Forming and Validating Causal Connections

This section describes the process to formulate cause and effect relationships into a chain of related events. It starts by iterating over all sub-rules $(c_x \implies c_y) \in C$ and combine the LHS event IDs of those sub-rules whose RHS event IDs are the same. This will output a group, *G*, containing the combined sub-rules. Each member $g \in G$ will have one or more event IDs on LHS linking a single event ID on RHS. The reason behind this combining process is to connect all events together, which lead towards the same goal. Although at this point, the combined events represent unordered steps to perform one or more particular actions. From the previous example, $g = (1234, 4567 \implies 5678)$, and in this case, $G = g$ as there are only two sub-rules in total. Set *G* implies that all events from LHS (1234, 4567) lead towards a single event on RHS (5678).

The next step is to create an ordered set of events within every $g \in G$, so that the sequence of cause and effect events can be determined. First, find those two event entries having a maximum time difference. It is assumed that those two events will mark the starting and ending events of action(s). Similarly, identify the second to last event based on the time that it happened before the ending event and so on. This process will arrange the event logs within *g* in terms of time, therefore defining the initial set of causal connections. Referring back to Figure 1, 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 is the starting point. Finally the wholesome causal connection rule (also referred as chain or sequence of events) would be (4567 $\implies$ 1234 $\implies$ 5678).

The next step is to find the truthfulness factor for each causal rule. Using the truthfulness factor, the causal connections will be arranged in terms of time. Considering the ongoing example, first, the truthfulness factor will be determined between 4567 and 1234 and then, 1234 and 5678. At any rule, if the resultant value is found to be lower than 50%, their locations will be swapped (replace cause with effect and vice-versa). As the swapping process will disturb the arrangement of an entire chain, the whole process will start from the beginning so that the swapped event is inserted into proper position in the sequence. This process ensures the correctness of causal relationships. Moreover, the average truthfulness value of each rule is also calculated to provide the validity of an entire chain.

4 EMPIRICAL ANALYSIS

This section presents empirical analysis of the developed technique using real-time event logs data. The event logs files are acquired from 10 different live machines under the university's network. The Apriori algorithm was used for extracting correlation rules from the event logs entries. A survey claims that different ARM algorithms show similar outputs despite being fundamentally different in terms of the employed strategy [2]. However, a study [8] shows that the itemset generation technique of Apriori is better in extracting complete and correct rules. It also uses the search space reduction non-monotonicity principle for better efficiency. This motivated the use of Apriori algorithm in proposed solution.

Due to large number of entries in a dataset and multi-step conversion process from correlation to causal relationship based rules, the minimum support (*minsup*) and confidence (*minconf*) values require careful selection. Different values were tested for *minsup* and *minconf* ranging from 10% – 50% and 20% – 90%, respectively. Finding the feasible values is an important for eliminating less interesting rules. For this paper, the values are chosen as *minsup* = 20% and *minconf* = 70%. These values are subjected to change due

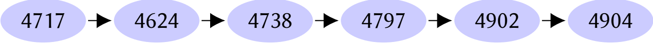

Figure 2: Causal relationship rule from dataset-10 with truthfulness value as 91.25%

Event logs dataset	Events entries	Unique entries	Correlation rules	Event-based rules	Sub-rules	Causal connections	Execution Time	Accuracy
1	7,756	33	13,423,448	23	47	7	1.8*m*	89
2	8,254	33	6,629,475	20	43	6	1.8*m*	78
3	11,956	29	281,508,612	24	64	10	4.5*m*	84
4	9,721	32	29,597,902	23	70	11	2.3*m*	67
5	10,206	31	27,839,778	24	69	11	2.4*m*	71
6	13,027	32	1,132,135	41	101	8	2.8*m*	93
7	10,948	15	12,472,706	4	16	5	2.4*m*	67
8	26,008	17	714,287	11	17	5	5.6*m*	80
9	3,404	33	114,728	15	259	18	0.7*m*	95
10	31,729	20	778,354	13	30	6	6.7*m*	100

Table 1: Results from performing empirical analysis during event acquisition and causal relationship mining processes.

to their dependency over the given dataset and frequency of its unique entries, however, they were found suitable for 10 event logs datasets collected from various sources. Future work will present more generalised values and their implications.

Figure 2 shows an example of an identified causal connection with a 91% truthfulness value extracted from dataset-10. It represents the sequence of events where an administrator searched for the password-less user accounts. Upon finding such an account, its management settings were changed. Following this, the user logged into the account and started performing certain activities.

The number of event logs, correlation rules and causal connections alongside execution time (minutes) and accuracy are presented in Table 1. The accuracy of causal connections is determined on the knowledge of two human experts. A certain rule was considered inaccurate (0) where both experts disagree, otherwise it was given half point (0.5) if one of the expert agree, and one point (1) if both of them agreed. Following conclusions are observed from the results:

(1) The number of causal connections are proportional to the number of unique events and their objects; and
(2) Number of correlation rules does not impact the number of causal connections and execution time.

It has been observed that datasets with lower accuracy contain such causal connections, where a single event is linked to many others. Another observation is that the solution does not create causal connections where they are not present, regardless of the number of entries in event logs. Four such datasets, originally part of the testing, were eliminated from this paper as they did not produce any causal connections. The object-based model ensures that a causal connection is created only where there is a relation among the properties of event log entries.

5 CONCLUSION

This paper presents a novel technique for automated extraction of causal connections from system event logs without any human assistance. The proposed technique is based on a scenario where an expert performs the security evaluation or configuration on a system, and every change is recorded in the form of event logs. Identifying causal connections among such event log entries provide sequences of actions that expert took to reform the system security. Those actions can be applied to other systems for increasing their security. The proposed solution was tested on 10 event logs datasets. Despite the event logs entries are produced in high frequency and might contain large amount noise, the proposed solution demonstrated that it can successfully extract the domain knowledge and be applicable in practical environments.

REFERENCES

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

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

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

[4] Yanqing Ji, Hao Ying, John Tran, Peter Dews, Ayman Mansour, and R Michael Massanari. 2013. A method for mining infrequent causal associations and its application in finding adverse drug reaction signal pairs. *IEEE transactions on Knowledge and Data Engineering* 25, 4 (2013), 721–733.

[5] Zhou Jin, Rujing Wang, He Huang, and Yimin Hu. 2014. Efficient interesting association rule mining based on causal criterion using feature selection. *JOURNAL OF INFORMATION &COMPUTATIONAL SCIENCE* 11, 12 (2014), 4393–4403.

[6] T Karthikeyan and N Ravikumar. 2014. A survey on association rule mining. *International Journal of Advanced Research in Computer and Communication Engineering* 3, 1 (2014), 2278–1021.

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

[8] B Nath, DK Bhattacharyya, and A Ghosh. 2013. Incremental association rule mining: a survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 3, 3 (2013), 157–169.

[9] Gregory Piatetsky-Shapiro. 1991. Discovery, analysis and presentation of strong rules. *Knowledge discovery in databases* (1991), 229–238.

[10] Kenneth J Rothman and Sander Greenland. 1920. Hill's criteria for causality. *Encyclopedia of biostatistics* (1920).

[11] Ramendra K Sahoo, Adam J Oliner, Irina Rish, Manish Gupta, José E Moreira, Sheng Ma, Ricardo Vilalta, and Anand Sivasubramaniam. 2003. Critical event prediction for proactive management in large-scale computer clusters. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 426–435.

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

[13] Derek Schauland and Donald Jacobs. 2016. Managing the Windows Event Log. In *Troubleshooting Windows Server with PowerShell*. Springer, 17–33.

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

[15] Judith D Singer and John B Willett. 2003. *Applied longitudinal data analysis: Modeling change and event occurrence*. Oxford university press.

[16] R Sumithra and Sujni Paul. 2010. Using distributed apriori association rule and classical apriori mining algorithms for grid based knowledge discovery. In *Computing Communication and Networking Technologies (ICCCNT), 2010 International Conference on*. IEEE, 1–5.

[17] Xiuli Yuan. 2017. An improved Apriori algorithm for mining association rules. In *AIP Conference Proceedings*, Vol. 1820. AIP Publishing, 080005.