Context-based Irregular Activity Detection in Event Logs for Forensic Investigations: An Itemset Mining Approach

Saad Khan[a,*], Simon Parkinson[a], Craig Murphy[b]

[a] *Department of Computer Science, School of Computing and Engineering, University of Huddersfield Queensgate, Huddersfield HD1 3DH, UK*
[b] *Defence Science and Technology Laboratory, Dstl Porton Down, Salisbury, Wilts, SP4 0JQ, UK*

Abstract

Event logs are a powerful source of digital evidence as they contain detailed information about activities performed on a computer. Forensic investigation of the event logs is a challenging and time-consuming task due to their large volume and continuous generation. A significant amount of time, effort, and knowledge is required to interpret their contents, discovering irregular events that are potentially pertinent to the investigation. As the number of digital investigations increases, so too must resources available to investigators. This requires new techniques to make the process easier and faster, reducing the burden on human investigators as well as being resource efficient. In this paper, a novel solution is presented to examine event logs and automatically identify irregular activities during forensic analysis. The proposed solution utilises a rare itemset mining approach to establish relationships among event entries, based on their contents. Following on, identified event relationships are ordered based on their temporal order to represent the timeline or sequence of activity. The solution is also capable of prioritising identified activities by calculating their degree of irregularity. The empirical analysis is performed on 15 live machines, and the results are discussed in terms of accuracy and performance metrics.

Keywords: Event Logs, Forensic investigation, Irregular activities, Rare itemset mining

1. Introduction

Event logs are an important source of information from the perspective of Digital Forensics and can be used as admissible evidence (Ibrahim et al., 2011). They contain detailed records of system operations and activities, based on system default and predefined auditing policies set by the administrator. Each entry is composed of a series of objects, which provide a contextual understanding of the event. Event logs are considered a keystone for understanding system operations, failure, performance, and security (Wang et al., 2019). They contain detailed state information on a system's controls and serve various purposes, including the identification of the problematic components of the system. For example, detecting irregular behaviour or activity of users and applications (Feremans et al., 2019b), malware attacks and their reconstruction (Chabot et al., 2015), identification of correlated/causal relationships among events (Khan & Parkinson, 2017), remote access and activity authentication activity (Sanjappa & Ahmed, 2017).

Digital forensic investigators are generally aware of what kind of information might be available in an event log and will have the necessary expertise and knowledge to discover information of interest. However, due to the large volume of events generated by modern computing systems, it is cumbersome and time-consuming for the expert investigators to perform an in-depth examination of all events with the view of determining their relevance to an investigation (Lillis et al., 2016). This challenge has resulted in researchers working on automating the process of knowledge discovery in event logs as it facilitates the quick and efficient acquisition of digital evidence without compromising the accuracy, reliability, soundness, and consistency that is provided by the human experts (Studiawan et al., 2019). Furthermore,

[*] Corresponding author
Email addresses: `saad.khan@hud.ac.uk` (Saad Khan), `s.parkinson@hud.ac.uk` (Simon Parkinson)

automation can help reduces the backlog of cases through better resource utilisation, helping to make timely progress in the investigation (James & Gladyshev, 2013).

Data mining is a process of extracting useful information from the data. As one of the common (automated) data mining methods for a number of applications, itemset mining is used to discover relationships among items in a given transactional database (Fournier-Viger et al., 2017). A wide variety of itemset mining algorithms are available that are suitable for detecting irregularities in the event logs under investigation. A popular method of knowledge discovery from event logs is to identify interesting patterns based on frequency. Recent examples of methods used to discover knowledge are: frequent itemset mining (Djenouri et al., 2018), association rule mining (Khan & Parkinson, 2018), causal relationships mining (Khan & Parkinson, 2017), and episode mining (Wu et al., 2013). Each pattern depicts a sequence of linked events, where each event is a record of the user, operating system, application software, or network activity on a computer. This pattern of events is used to define an *activity*. Besides extracting knowledge based on frequent patterns, there is another approach based on mining *rare* patterns that do not occur frequently in the dataset. Based on a premise that malicious activities do not occur frequently in a system, mining interesting rare patterns can provide valuable insights to the investigator in terms of identifying hidden and potentially significant irregular, suspicious, or anomalous activities, within a reasonable amount of time (Darrab et al., 2021; Parkinson & Khan, 2018). Rare mining is further justified because if events of interest to an investigation were occurring frequently, then they would be easily and quickly detected by the investigator. In this paper, define an *irregular activity* as being a sequence of events that are determined to occur infrequently due to their contents. More specifically, the occurrence of objects in the events.

To motivate further, consider a hypothetical scenario where an investigator has the task to forensically analyse the event log of a network file server to try and discover any evidence related to the case. It is suspected that an unauthorised user gained access to, viewed, and copied a sensitive document. Initially, the investigator notices that a local user account, named `Bob`, sent a request to access the file-share service on the server (Microsoft event ID*5145*[1]). Next, an event is logged that shows the server accepted the request and allowed access to `Bob` (ID *5140*). Following on, the server asked `Bob` for the login credentials, and an event was lodged to record the login action (ID *4648*). Upon the successful login, the investigator discovers an event entry that showed a record of `Bob` adding a new registry value (ID *4657*). The next two events reveal that `Bob` logged out of the server (ID *4647*) and then logged in again (ID *4624*). However, after examining the login event (ID *4624*), it was discovered that the second login was performed with a different group security identifier, which allowed a higher level of access to the server. This demonstrates that `Bob` compromised the server most likely by exploiting a privilege escalation vulnerability in the file-share service and adding a value in the registry for gaining higher access permission. Identifying this activity registry modification activity by a local user account is a case of a malicious insider. Such malicious activities are not as frequent as benign activities of the system as those recorded resulting from system operations that are happening irrespective of user operation. The event logging mechanism records all types of activities, which is why it requires a considerable amount of time and effort to manually find irregularities in the large-scale event logs. An automated solution that can detect such rarely occurring, suspicious event patterns would improve the efficiency and productivity of the forensic analysis process. Therefore, the aim of this paper is to investigate and develop an automated mechanism to identify irregular activities from event-based data sets.

In this paper, we present a mechanism to identify irregular or anomalous system event entries. The following list presents our main contributions:

1. A technique to reduce the number of rare itemsets generated by existing rare itemset mining algorithms for further processing. This increases the performance of the forensic investigation process, without losing any information in the itemsets;

2. An algorithm to discover and quantify repetitions of event patterns. The algorithm is utilised to prioritise rare event patterns for the forensic investigator for analysis based on their degree of irregularity; and

3. Finally, a software application for processing Microsoft event logs and presenting the identified irregular event patterns.

[1]The full description of each Microsoft event can be found at the following URL: `https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventid=5145`

The document is organised as follows: Section 2 reviews existing approaches for irregular activity detection in the event logs that can be useful in forensics investigations, highlighting the need and importance of the proposed system. Section 4 explains the data preparation phase of the proposed system. Section 5 describes the process of mining and filtering rare itemsets. Section 6 demonstrates the proposed mechanism of transforming the itemsets into sequences of event entries, where each sequence represents a single irregular activity. Finally, Section 7 presents the empirical analysis of 15 live machines to demonstrate the performance and accuracy of the proposed solution.

2. Related Work

A recent paper has presented a solution to produce human-understandable summaries of an event log (Dijkman & Wilbik, 2017). The summary is a textual representation of event patterns discovered by using clustering techniques. The solution was evaluated by 51 participants and they found the summarises useful and easy to understand. Another paper presents a solution to mine knowledge from event logs, called FLAP (Li et al., 2017). The solution utilises manual rule-based and automated template learning techniques to extract meaningful information from unstructured event logs, as well as perform event summarisation, and failure prediction amongst other analytic approaches. The authors claim that FLAP has been successfully deployed in a live system and has proven effective in establishing the daily behaviour of the system, enabling the identification of irregular activity, and assisting in identifying the root causes of system failures. Recent work has proposed an automated event log abstraction technique to aid forensic investigations (Studiawan et al., 2020). The technique starts with building a graph model based on word frequency. The graphs are then automatically clustered into the best configuration, before merging into compact abstractions. The developed solution was tested on five datasets, performing better than existing solutions on all datasets by obtaining the highest mean F-measure of 95.35%.

Another research study presents ADELE (Khatuya et al., 2018), a framework that utilises enterprise application event logs to predict system failure and automatically alert the authorities. The framework starts by learning a model of known failures using Ridge regression based on 18 feature attributes, such as event count, mean time between successive events, severity, and others. The model is used to classify the incoming events into normal or abnormal based on a scoring system. The novelty of ADELE lies in several aspects: its capacity to discriminate between failure and irregularity, identification of the subsystem that is causing the failure, as well as performing short-term live failure predictions. The evaluation of ADELE was performed on a small dataset containing 4,800 entries, demonstrating an 83% true positive rate and 12% false positive rate. In other related work, the authors propose the DeepLog framework that models system log data into a natural language sequence using a combination of deep learning and clustering algorithms (Du et al., 2017). DeepLog can be utilised in several applications and environments. It starts by identifying activities in the log data and then creates a workflow model, before identifying any deviation from the model as an anomaly. The solution was tested on two large datasets (Hadoop and Openstack) and achieved the best overall performance with an F-measure of 96%.

Process mining is a technique to automatically learn knowledge from event logs. It has three main areas Kalenkova et al. (2017): process discovery (constructing process models/workflow from event data to gain insight), conformance checking (finding deviations or anomalies from desired system behaviour), and enhancement of process models with contextual information. According to an extensive survey dos Santos Garcia et al. (2019), process mining has been successfully applied in several domains (primarily for process discovery), such as healthcare, education, finance and many others. One research study Hendricks (2019) used ProM (Process Mining Framework Van Dongen et al. (2005)) to analyse the hospital event log of sepsis patients brought into the emergency room to understand and streamline clinical operations. The analysis identified several processes that can be refined to reduce patient wait times and re-admissions; thereby, reducing staff workload and costs. Another research study De Leoni et al. (2016) developed a generalised process mining framework that uses correlation and clustering techniques for process discovery from the event logs. The extracted knowledge is represented as decision rules, which are used for the root cause analysis of observed problems and to provide novel insights into running instances of the process model. Another article Jans et al. (2014) introduced a new type of auditing technique for banking transactions using process mining on its event logs data. The evaluation was performed on actual data from one of Europe's largest banks. The technique identified numerous transactions where payments were made without approval, and also those that violated internal procedures. It also discovered anomalies that went undetected by the bank's internal auditors. A similar, more recent study Chiu & Jans (2019) also demonstrated the usefulness of process mining of audit-relevant event logs to detect key issues, potential risks, and poor internal controls and processes. Another article Bahaweres et al. (2021) utilised PRoM on finance-related, real-life event logs for forensic auditing and deemed process mining an effective analysis method to detect fraud. The authors also claim that the analysis can assist in improving processes by identifying unusual activities to prevent financial loss.

One recent paper has proposed a clustering algorithm to determine outlier event patterns in log data named, LogCluster (Vaarandi et al., 2016). The algorithm operates in a fully automated manner, although, it requires the user to input a support threshold value that defines the least number of events in each cluster. The algorithm employs two heuristic schemes to generate more comprehensible and quality results. In general, the existing clustering-based irregularity detection techniques are static in nature. They create clusters of similar events based on similarity metrics and present dissimilar events as irregular; however, it would require a complete reformation of clusters if the underlying technical infrastructure is changed. To resolve this issue, in one study the researchers propose a novel clustering mechanism for detecting anomalous events in a dynamic fashion (Landauer et al., 2018). This mechanism builds static clusters and uses a temporal metric to arrange them in sequence. It then analyses neighbouring clusters that are within a pre-defined time window to extract evolution metrics, such as cluster growth rate, change rate, and stability rate. An autoregressive integrated moving average model is then trained to conduct time-series analysis, determining links and transitions between seemingly isolated clusters. Finally, incoming data is classified as irregular or regular using one-step ahead forecasting, which is performed using extrapolation to determine whether the incoming data is as expected by the model or not.

The clustering-based techniques, albeit useful for forensic investigators in obtaining a summary view of the event log entries (generally large in number), might find similar events that are not necessarily related to each other (or generated by a single user, application, network, or system activity). This issue can be resolved by employing pattern mining techniques. For example, a research study proposed a new solution for detecting anomalies in SCADA event logs using a Rare Sequential Pattern Mining (RSPM) technique (Rahman et al., 2016). This solution requires user input to define support values for mining patterns. The solution was tested on publicly available SCADA logs and has shown reasonable results. In another work, the authors use Probably Approximately Correct (PAC) learning to detect irregularities in any data by finding rare patterns in an unsupervised manner (Siddiqui et al., 2016). Every pattern in the selected pattern space is considered a hypothesis (for input). After performing probability estimation, all those patterns that have a lower probability than the user-defined threshold are marked as irregular. A pattern-based technique has also been applied to determine irregularities in mixed-type time series data that contains both continuous (from sensors) and discrete values (Feremans et al., 2019a). Initially, all frequent patterns are acquired to determine normal behaviour and then mapped onto the input data using similarity metrics to obtain certain feature vectors. An unsupervised algorithm called, isolation forest, is utilised to identify irregularities as they have relatively shorter path lengths in the tree representation.

Rare pattern mining has also been applied in areas outside of computer security. For example, in one study, the authors analysed e-learning data by applying Rare Association Rule Mining (RARM) algorithm to aid tutors in identifying infrequent student behaviour and provide learning support (Romero et al., 2010). The evaluation data is collected from 230 students in five courses and the results show several interesting relationships between online activities (E.g., quizzes and forum participation) and the final mark obtained by the students. Another application presents a method for finding rare itemsets in weblog data to analyse the behaviour of user's click path (or clickstream), along with identifying rare web pages (Bakariya & Thakur, 2016). The proposed method is based on a power set lattice traversal technique and is claimed to take less computational time and resources in comparison with existing algorithms. Another recent work provides a framework to automatically discover case notions within the large-scale event log, using a combination of regression and learning to rank (LTR) algorithms (de Murillas et al., 2020). A case notion is defined as a mechanism to group correlated events from a specific perspective. The case notions are built using features, such as the maximum number of events and the total number of events. The system also provides recommended case notions to the user based on certain interestingness-related metrics, aiming to save time and effort over manual analysis. There are several other research works that use rare patterns to determine anomalies in different conditions, such as anomaly detection under uncertainty (Islam et al., 2018) and identifying anomalies in large-scale network traffic data (Ahmed, 2016).

According to an extensive survey, rare pattern mining has been applied to resolve a wide range of real-life problems and challenges. Most prominent application areas include network intrusion detection, credit card fraud detection, medical diagnosis, insurance risk modelling, web mining and hardware fault detection (Borah & Nath, 2019). There is an argument that rare (not recurrent) events can be considered as irregular as they do not conform to the normal system behaviour, profile, or execution, and their analysis can play a crucial role in identifying cyber-attacks and system failures (Rahman, 2019). Therefore, in this paper, we propose a rare itemset mining-based technique to determine irregular or anomalous activities in any operating system without any human intervention, using the default event logs generated and stored by the operating system. To demonstrate the application and suitability of the technique, we have used Microsoft's Event log sources. An important aspect of our proposed solution is unsupervised learning that does not require large and labelled training data or a significant amount of time and computing resources. The developed technique outputs rare sequences of event log entries that can assist any digital forensic investigator to perform quick, context-aware identification of potentially suspicious activities, without having any kind of prior knowledge about the system or spending huge amount of time, effort and resources for manual analysis.

3. Summary of the proposed solution

The following list provides an overview of the steps that are included in the developed technique (also shown in Figure 1). The purpose of this list is to provide a summary and highlight the novelty of the approach to the reader before the details of each aspect are described in detail in separate sections.

1. *Data preparation* – Acquire and represent Event log data into a uniform format, referred to as an object-based model, which is structured and efficient for processing;
2. *Closed Rare itemset mining* – Generating rare itemsets from the object-based model within a specific support limit and then converting them into a compressed, more compact representation by removing all non-closed rare itemsets; and
3. *Detecting Irregular Activities* – Using the closed rare itemsets to discover sequence or patterns of related event entries, which were triggered due to particular system activity, along with prioritising the patterns based on their level of rarity.

The main idea behind our research is to develop a high-performance tool that can decrease the workload and effort of human forensic investigators when compared to manual analysis in discovering irregular patterns in large-sized event logs. Following are the contributions of this work along with briefly describing the purpose of each step:

Efficient event log data representation – The first part aims at preparing the event log data for the itemset mining algorithm and begins by retrieving the event log from a machine that requires forensic investigation. All entries in the event log are parsed to extract key objects and assigned sequence identifiers. Following on, common objects within entries that have the same event type are identified and removed, so that the remaining objects are smaller in size and yet better representatives of their corresponding entries due to the increased level of uniqueness amongst them. This concludes the first stage, generating an object-based model of the given dataset.

Reduced number of rare itemsets for consideration and analysis – In the second stage, the object-based model is fed into an unsupervised algorithm to generate the rare itemsets using a wide-ranging 1%-50% support threshold. This also helps in reducing the probability of human error without needing significant prior knowledge of the given event log. The next step in this stage is to eliminate all non-closed rare itemsets and formulate their compressed representation to elevate the solution's performance and efficiency, without compromising any knowledge contained in the rare itemsets.

Determining specific event entries involved in irregular patterns and prioritising found activities – In the final stage, every closed rare itemset is converted into a context-based pattern by obtaining the matching event entries from the original dataset, which can also help in further or more detailed forensic analysis. The entries within every context-based pattern are temporally ordered based on their corresponding sequence identifiers (or timestamps) to present a complete (possibly) malicious activity. Finally, every activity is assigned a degree of irregularity in the form of a numeric value for the forensic investigator to show how much rare is this particular pattern and which patterns are deemed important by the tool and should be considered first.

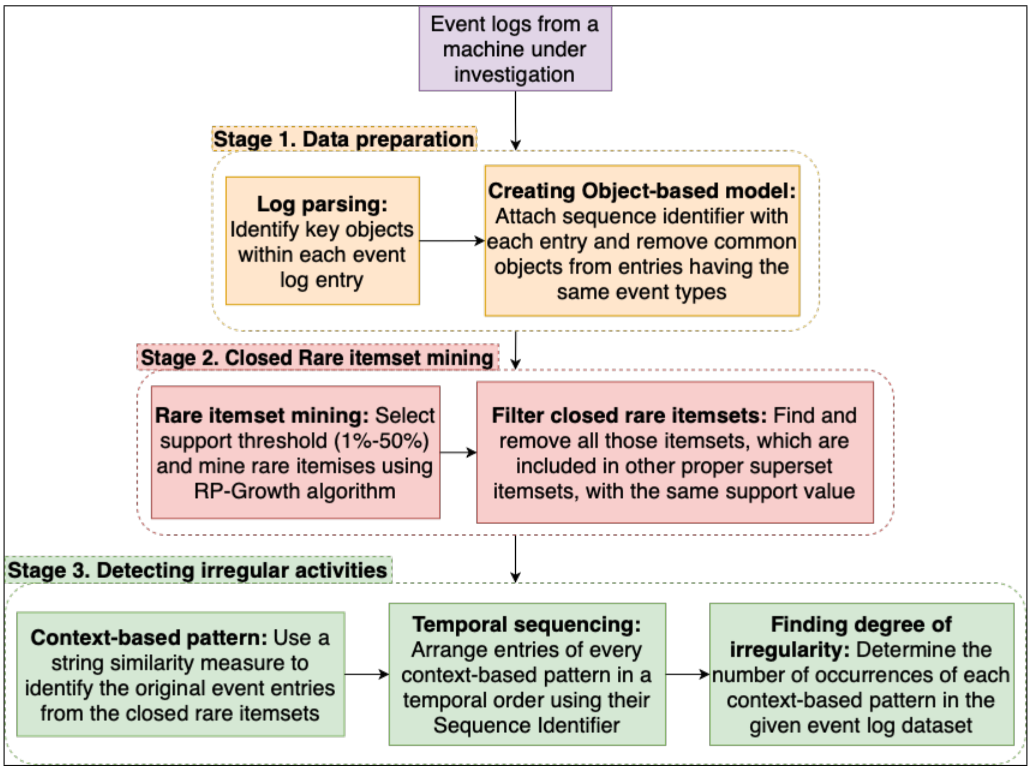

Event logs from a machine under investigation
Stage 1. Data preparation
Log parsing: Identify key objects within each event log entry
Creating Object-based model: Attach sequence identifier with each entry and remove common objects from entries having the same event types
Stage 2. Closed Rare itemset mining
Rare itemset mining: Select support threshold (1%-50%) and mine rare itemises using RP-Growth algorithm
Filter closed rare itemsets: Find and remove all those itemsets, which are included in other proper superset itemsets, with the same support value
Stage 3. Detecting irregular activities
Context-based pattern: Use a string similarity measure to identify the original event entries from the closed rare itemsets
Temporal sequencing: Arrange entries of every context-based pattern in a temporal order using their Sequence Identifier
Finding degree of irregularity: Determine the number of occurrences of each context-based pattern in the given event log dataset

Figure 1: Overall process of the proposed solution.

```
Permissions on an object were changed.
     ID: log ID 4670
Subject:
     Security ID:    Admin
     Account Name:   Clive
     Account Domain: AD
     Logon ID: 0x9B3EC
Object:
     Object Server:  Security
     Object Name:    D:
     Handle ID:      0x5bc
Process:
     Process ID:     0x1820
Permissions Change:
     Original Security Descriptor:
D:(A;OICI;FA;;;SY)(A;OICI;FA;;;BA)
     New Security Descriptor:
D:ARAI(A;OICIID;FA;;;SY)(A;OICIID;FA;;;BA)
(A;OICIID;FW;;;bob)
```

Figure 2: Example Microsoft event (ID 4670) detailing the change in security permissions

```
Apr 28 17:06:20 ip-172-31-11-241 sshd[12547]: Invalid user admin from 216.19.2.8
```

Figure 3: Example Syslog event for an invalid user login

4. Data preparation

4.1. Log parsing

A typical system for mining patterns from event logs begins by parsing both structured and unstructured logs to normalise and standardise their format, along with extracting meaningful usable parts, such as key-value pairs. A recent survey details existing tools and benchmarks for automated log parsing (Zhu et al., 2019). The presented approach is motivated by demands on the forensic analyst within the Microsoft Windows environment, yet is capable of handling unstructured data. Microsoft Windows is one of the most commonly found operating systems during forensic investigations. Furthermore, the events are structured in the form of object key-value pairs, which makes it easier to parse. However, the proposed solution is general and applicable to different formats. An example Microsoft Event detailing a change in security permissions is provided in Figure 2. The example provides an illustration of key-value pairs in a structured event. For example 'Account Name' is the key and 'Clive' is the value. An example of an unstructured event is shown in Figure 3. As there are no key-value pairs, all words of the log message 'invalid user admin from 216.19.2.8' are processed as objects, and where possible, multiple similar entries are compared to remove descriptive text. For example, a comparison of the two events of the type shown in Figure 3 would enable the easy identification of the words 'invalid', 'user', and 'from' as being descriptive parts of the text.

In the following discussion, D is used to represent the entries of the Microsoft Windows event log dataset acquired from a single computer, $D = \{e_1, e_2, ..., e_N\}$. Each event entry, $e = \{T, TS, O\}$ consists of an event type T, a timestamp TS, and a set of event objects O. Each event's object set O belongs to a global set of all objects, $I = \{o_1, o_2, ..., o_n\}$, such that $O \subseteq I$. Each entry, e_i, contains corresponding objects from I to represent an occurrence of an event. The objects could be user name, system resource and permission level. For example, consider a simple event log dataset $D_1 = \{e_1, e_2, e_3, e_4, e_5, e_6\}$, where:

$$
\begin{aligned}
e_1 &= \{\{ET1\},\{1/1/20; 00:00:01\},\{A, B, C, D\}\} \\
e_2 &= \{\{ET2\},\{1/1/20; 00:00:01\},\{A, E, D\}\} \\
e_3 &= \{\{ET1\},\{1/1/20; 00:00:10\},\{B, E, A, F\}\} \\
e_4 &= \{\{ET3\},\{1/1/20; 00:02:09\},\{A, F, E\}\} \\
e_5 &= \{\{ET4\},\{1/1/20; 01:03:49\},\{B, F\}\} \\
e_6 &= \{\{ET5\},\{1/1/20; 01:03:52\},\{D, F, C\}\}
\end{aligned}
$$

In this trivial example, each entry contains an event type, timestamp, and set of objects. It is worth noting that in this work, an activity is a set of linked events. Note that the original event log contains a large number of entries that are more diverse due to containing a record of several system operations and each entry contains many objects. Events occur over the duration of the system running and they generally appear in sequence; however, the concurrency of different processes can generate intricate entries.

4.2. Creating object-based model

After parsing the event log dataset, the next step is to formulate event log entries into an object-based model (OBM), where each entry is represented in terms of the system objects described in the event log entry. The main purpose of OBM is to convert any given event log dataset into the same input format that is required for further processing. Building the OBM requires the following two steps: attach a sequence identifier (ID) with each entry and remove common objects from entries having the same event type.

For the first step, all entries are sorted in an ascending order list (i.e., oldest to latest) with respect to TS. Some types of event datasets, E.g., Windows event logs, contain a unique record or sequence number by default. Such record numbers can be used as sequence IDs. In the second step, all entries that have the same event type are grouped together and their common objects are eliminated. The purpose of these two steps is to increase the level of uniqueness among the entries of the same event type (i.e., each entry of the dataset is uniquely identifiable), along with improving the accuracy of context-based itemsets in Section 6.1.

After processing the event log entries based on the aforementioned steps, the object-based model of D is represented as $OBM_D = \{E_1, E_2, ..., E_N\}$. The event entry becomes $E = \{SID,T,TS,O\}$, where SID is the newly assigned sequence ID of the event, T is the event type, TS is the time stamp, and O is the remaining set of useful objects associated with the event. Continuing our example of D_1, there are two entries (e_1 and e_3) having the same event type ($ET1$), and after removing their common objects (A and B), the $OBM_{D_1} = \{E_1, E_2, E_3, E_4, E_5, E_6\}$ is presented in the following:

$$E_1 = \{\{1\},\{ET1\},\{1/1/20; 00:00:01\},\{C, D\}\}$$
$$E_2 = \{\{2\},\{ET2\},\{1/1/20; 00:00:01\},\{A, E, D\}\}$$
$$E_3 = \{\{3\},\{ET1\},\{1/1/20; 00:00:10\},\{E, F\}\}$$
$$E_4 = \{\{4\},\{ET3\},\{1/1/20; 00:02:09\},\{A, F, E\}\}$$
$$E_5 = \{\{5\},\{ET4\},\{1/1/20; 01:03:49\},\{B, F\}\}$$
$$E_6 = \{\{6\},\{ET5\},\{1/1/20; 01:03:52\},\{D, F, C\}\}$$

5. Closed Rare itemset mining

5.1. *Rare itemset mining*

The primary aim of itemset mining is to extract interesting and useful groups of items (itemsets) from the dataset that are representative of actionable knowledge. The interestingness of an itemset is usually measured by *support*, which is defined as the number (or ratio) of transactions in the dataset containing the itemset. Using the right support is essential to producing high-quality rules, according to a study of 61 different interestingness measures on 110 datasets (Tew et al., 2014).

Frequent itemsets have higher support than a user-defined threshold and identify recurrent activities or regularities. On the other hand, rare itemsets have lower support and correspond to unexpected activities or irregularities, possibly confirming or contradicting one or more informed suppositions made in the dataset. Several approaches have been developed for rare itemset mining, mainly due to different constraints on what constitutes a rare itemset and conditions to prevent discovering irrelevant and large numbers of rare itemsets. For example, perfectly rare itemset using AprioriInverse (Koh & Rountree, 2005), minimal rare itemset using AprioriRare (Szathmary et al., 2007), and mining all rare itemsets RPGrowth (Tsang et al., 2011) algorithm.

Most existing approaches take user-defined support, which can be difficult for the investigator to provide due to a lack of prior knowledge about the dataset. However, some methods are also available that can automatically compute the support threshold. (Sadhasivam & Angamuthu, 2011) presents an exhaustive approach that extracts all possible rules from a dataset and then determines the average and median of all support values of the rules to filter the interesting rules. Other studies determine the support threshold based on the frequency analysis of the unique items in the dataset, using measures of tendency and dispersion (Selvi et al., 2016; Khan & Parkinson, 2019).

5.1.1. *Support threshold*

The proposed solution primarily aims at performing an in-depth analysis of event logs, whilst reducing the manual effort required during a forensic investigation. Therefore, all possible rare itemsets in the support range of 1% to 50% are processed. Although an intensive approach, it will prevent the investigator from missing any interesting pattern, which is the main purpose of this study. This trade-off between identifying all patterns and consumption of computing resources can lead to a successful investigation as one or more extracted patterns might constitute digital evidence. Note that the sequence identifiers, event types, and time stamps are not processed during rare itemset mining. Only the event's objects are processed. In later stages, the proposed solution applies additional filtering mechanisms to highlight the most interesting patterns based on their frequency of occurrence in the event logs.

Entries	Support	Algorithm	Execution time (ms)	Number of Rare itemsets
6	1%–50%	AprioriRare	2	6
		AprioriInverse	3	10
		RPGrowth	3	15

Table 1: Comparison results for the rare itemset mining algorithms using the running example based on the number of generated itemsets.

5.1.2. Algorithm selection

As previously discussed, several rare itemset mining algorithms are available that can be applied to event log data, each having different strengths and weaknesses. For this paper, we considered the following three algorithms: RPGrowth, AprioriInverse, and AprioriRare. The proposed solution employed RPGrowth for two main reasons. First, as demonstrated in the results from executing all three algorithms on the running example OBM_{D_1} in Table 1, RPGrowth generates a higher number of rare itemsets (15) in the time span of three milliseconds. This is due to the fact that RPGrowth is an extension or equivalent of FPGrowth algorithm, which is why it inherits the speed, efficient memory consumption, mining informative itemsets, and several other benefits by using the compact data structure called FP-Tree (Ranjan & Sharma, 2019). Other algorithms produced a lower number of rare itemsets within an approximately similar amount of time. We argue here that extracting most (or all) rare itemsets means that there is less chance of missing any itemsets that might contain critical or useful information in regard to the investigation. Second, both AprioriInverse and AprioriRare algorithms suffer from the same deficiencies as the Apriori approach in generating candidate itemsets (Garg & Gulia, 2015), which renders them infeasible for large-scale datasets and practical use.

To further demonstrate the suitability of the selected algorithm, we execute all three algorithms on the 15 evaluation datasets. The datasets and evaluation procedure are explained in Section 7.1. The same trend is observed in these datasets (some results are shown in Table 2), where RPGrowth outperformed AprioriRare and AprioriInverse algorithms in terms of the number of mined rare itemsets. Furthermore, the time consumed by RPGrowth for each dataset, where the number of log entries varies between 522 and 69,856, is not significantly different than the time consumed by other algorithms, considering the number of output itemsets by each algorithm. Hence both Table 1 and Table 2 illustrates that RPGrowth algorithm is a suitable choice for the developed solution.

5.2. Filtering closed rare itemsets

Given the exhaustive nature of the selected support threshold, the proposed solution generates a large number of rare itemsets. This makes further processing inefficient and infeasible for practical use due to large time and memory consumption, and the final results might contain many redundant or irrelevant itemsets (Zaki & Zulkurnain, 2018). To overcome this issue, the proposed solution identifies and removes all non-closed itemsets from the rare itemsets (extracted in the previous section), and outputs $R = \{r_1, ..., r_n\}$, where $r = (o_1, o_2, \ldots, o_N)$, and o is the object value from an event entry.

This component of the solution is inspired by the concept of closed frequent itemset mining (Martin et al., 2020). An itemset is considered closed if none of its immediate supersets has the same support as the itemset. In other words, a closed itemset is an itemset that is not included in any other itemset having exactly the same support. The reason behind removing all non-closed rare itemset is to reduce the size of itemsets without losing any useful information contained in them. The closed rare itemsets are the compressed representation of the rare itemsets, i.e., they can regenerate all original rare itemsets along with the correct support values.

Algorithm 1 presents the developed technique to identify and remove all non-closed itemsets from the rare itemsets found by the RP-Growth algorithm. The algorithm has a complexity of $O(N^2)$. The algorithm starts by repeatedly iterating through the acquired rare itemsets using the nested loops on line 2 and line 3. It then compares each pair of adjacent items at i^{th} and j^{th} indices to determine whether (1) one is a proper subset of another, and (2) both itemsets have the same support. If any condition is satisfied on line 4 or line 6, the rare itemset on the i^{th} or j^{th} index is removed by line 5 or line 7, respectively.

After extracting the rare itemsets from the event log dataset using the RPGrowth algorithm and removing all non-closed rare itemsets using Algorithm 1, Table 3 shows the remaining amount of closed rare itemsets for the running example OBM_{D_1}. The initial set contained 12 rare itemsets, whereas, the final set contained only six closed

Log	Entries	Algorithm	Execution time (ss.ms)	Rare itemsets
1	5028	AprioriRare	00.5485343	5445
		AprioriInverse	00.6308584	51
		RPGrowth_itemsets	**00.7749747**	**70033**
2	8254	AprioriRare	00.6187170	17977
		AprioriInverse	00.8159114	106
		RPGrowth_itemsets	**01.1114426**	**154498**
3	522	AprioriRare	00.5471202	789
		AprioriInverse	00.5362371	95
		RPGrowth_itemsets	**00.6190369**	**22024**
4	1123	AprioriRare	00.5614736	1645
		AprioriInverse	00.6089543	101
		RPGrowth_itemsets	**00.6359820**	**19753**
5	1080	AprioriRare	00.5228517	1661
		AprioriInverse	00.5722177	105
		RPGrowth_itemsets	**00.6402766**	**17732**
6	1692	AprioriRare	00.5459580	2500
		AprioriInverse	00.5579049	86
		RPGrowth_itemsets	**00.5974411**	**13889**
7	1715	AprioriRare	00.5691586	2457
		AprioriInverse	00.5998506	110
		RPGrowth_itemsets	**00.6606997**	**21741**
⋮				
15	15642	AprioriRare	00.6692261	24722
		AprioriInverse	00.8797968	56
		RPGrowth_itemsets	**00.8309945**	**56479**

Table 2: Comparison results of the rare itemset mining algorithms with 1%–50% support using the evaluation datasets. RPGrowth generates a higher number of rare itemsets than the AprioriRare and AprioriInverse algorithms within a similar time span.

Algorithm 1 Identify and remove non-closed rare itemsets.

```
Input: Set of all rare itemsets $R$
Output: Set of all closed rare itemsets $R$
 1: procedure REMOVE ALL NON-CLOSED RARE ITEMSETS
 2:     for all $r_i \in R$ do
 3:         for all $r_{j=i+1} \in R$ do
 4:             if $r_i \subset r_j$ and Support($r_i$) = Support($r_j$) then
 5:                 $R = R \setminus r_i$
 6:             else if $r_j \subset r_i$ and Support($r_j$) = Support($r_i$) then
 7:                 $R = R \setminus r_j$
 8:             end if
 9:         end for
10:     end for
11: end procedure
```

rare itemsets. For example, the rare itemset $\{B\}$ is a subset of $\{B,F\}$ with the same support, hence removed from further processing. Similarly, $\{C\}$ and $\{A\}$ are eliminated as they were found in $\{C,D\}$ and $\{A,E\}$, respectively, with the same support value of two. The itemsets $\{C,F\}$ and $\{D,F\}$ were also removed as both appeared in $\{C,D,F\}$, having the same support. Finally, all $\{A,F\}$, $\{A,D\}$ and $\{D,E\}$ are proper subsets of either $\{A,E,F\}$ or $\{A,D,E\}$ with the same support value, hence they were removed. It is evident that a significant amount of itemsets have been removed for every dataset without losing any information. This demonstrates the feasibility of the proposed solution in practical environments, such as forensics investigations, where both speed and accuracy are of vital importance.

No.	Rare itemsets	Support	Closed rare itemsets
1	B	1	×
2	B, F	1	B, F
3	C	2	×
4	C, F	1	×
5	C, D	2	C, D
6	C, D, F	1	C, D, F
7	A	2	×
8	A, F	1	×
9	A, D	1	×
10	A, E	2	A, E
11	A, E, F	1	A, E, F
12	A, D, E	1	A, D, E
13	D, E	1	×
14	E, F	2	E, F
15	D, F	1	×

Table 3: Reduction in the number of extracted rare itemsets after removing all non-closed rare itemsets by the solution.

6. Detecting Irregular Activities

6.1. Context-based pattern

At this stage, closed rare itemsets have been mined using the event objects. The itemsets only represent the rare relationships amongst the event objects, which might not provide enough useful analysis/information for forensic investigation purposes. Therefore, the next stage is to utilise the (object-based) itemsets to determine rare relationships amongst their corresponding event entries in the dataset. The related event entries and the information they contain will provide a useful, more specific context (or insightful view) of the irregular system activities, without any human intervention, thus saving time and resources. Also, this in-depth understanding of irregular (possibly suspicious or malicious) activities will enable the investigator to conduct further investigation as required.

Using closed rare itemsets to find connected event entries is performed by matching the objects of a closed rare itemset with the objects of all event entries. On finding a complete match with an entry, both the event type and the sequence identifier are extracted. This way, every closed rare itemset is formulated into a set of event entries that are linked to each other depicting a system activity, referred to as a context-based pattern of events. The term 'context' is used to highlight the fact that the acquired patterns of events are not universal, these relationships are specific and only exist in the given event entries. The context-based patterns are denoted as set $P = \{p_1, p_2, ..., p_N\}$, where $p = \{E_1, E_2, ..., E_N\}$ and E is the matched event entry. It has also been observed that the proposed solution might generate duplicate context-based patterns, which are immediately excluded from further processing.

6.2. Temporal sequencing

At this stage, it is clear that the relationships among events exist as the related event entries are grouped together in their respective context-based patterns. However, the crucial information of ordering in event entries, i.e, which event entry occurred first, second, and so on, is absent. It is important to identify the order of entries as it will inform the investigator about the exact sequence of events (or in other words, step-by-step execution of an irregular activity) that happened on the underlying machine.

The proposed solution utilises the previously assigned sequence identifiers, which are based on the timestamps of events, to facilitate the ordering. The order of event entries in each context-based pattern is performed by sorting sequence identifiers in ascending order. In other words, the ordering of event entries is determined based on when the events were triggered and logged.

After the temporal sequencing process, each context-based pattern in P is represented as $p = \{E_1 \rightarrow E_2 \rightarrow ... \rightarrow E_N\}$, where E is the precise event entry involved in a certain rare activity and the sequence/order of event relationship is defined by ($\rightarrow$) symbol. Note that similar approaches have been employed to improve the quality and efficiency of various pattern mining algorithms in time-series data (Batal et al., 2013).

6.3. *Finding degree of irregularity*

A professional or home computer might contain a record of multiple anomalous or malicious activities in its event log, therefore, the proposed solution will also identify multiple (rare) event patterns. All extracted patterns are unique in terms of events and their relationships, so we believe that it will be beneficial for the investigator to automatically prioritise the patterns based on how rare they are. To achieves this, a 'degree of irregularity' measure is used and it is represented as a numeric value. This will enable the investigator to inspect rare or irregular patterns first, thereby, making the solution suitable and applicable in a real-world environment.

To prioritise the context-based activity patterns with respect to the degree of irregularity, we have devised algorithm 2 to determine the number of occurrences of each pattern in the given event log dataset. This algorithm is based on a function that takes two arguments as input: (1) a pattern of event types $p_{ET} = \{ET_{E_1} \rightarrow ET_{E_2} \rightarrow ... \rightarrow ET_{E_N}\}$ extracted from each pattern p found in the set of temporally sequenced context-based patterns P, where ET_E represents the event type of the corresponding entry E and (2) the object-based model of event entries (OBM_D). The algorithm outputs a whole number to indicate the number of times a certain pattern of event types p_{ET} occurred in OBM_D.

The algorithm starts by initialising two variables, i and *PatternCounter*, with zero value on line 2 and line 3, respectively. The variable i is a loop counter to iterate over the event entries in OBM_D, whilst, the variable *PatternCounter* is used to count and store how many times the given pattern occurred in OBM_D. The iteration over event entries is performed by a while-loop on line 4. Another two variables are initialised inside this while-loop, j and $TempCounter$, with zero value on line 5 and line 6, respectively. The variable j is a loop counter to iterate over the pattern (p_{ET}) elements, whilst, the variable $TempCounter$ shows how many event types of p_{ET} have been successfully found in OBM_D.

The iteration over p_{ET} is performed by a nested while-loop on line 7, where every element of p_{ET} is searched in OBM_D. All event types on the i^{th} index of OBM_D and the j^{th} index of p_{ET} are matched using an if-condition on line 8. The value of both loop counters (i and j) are incremented by one on line 9 and line 12, respectively, after matching the elements of p_{ET} and OBM_D on the corresponding indices. Upon finding each match, the value of $TempCounter$ is incremented by one on line 10. After processing all event types of p_{ET}, if the value of $TempCounter$ is same as the number of events in p_{ET} on line 14, i.e., if all event types of p_{ET} are successfully mapped in OBM_D, the value of *PatternCounter* is incremented by one on line 15. In this manner, the occurrence of p_{ET} is repeatedly counted until the algorithm processes all entries of OBM_D, and the final value of *PatternCounter* is returned by line 18. This value can range from $1 - N$, where 1 represents the most rare pattern.

Algorithm 2 determines the frequency of occurrence for every context-based pattern in the event entries. Following on, all patterns are arranged in the order of least frequent to most frequent, as it is probable that the pattern with the least frequency is the most irregular. This algorithm also assists the investigator in discarding the insignificant patterns based on their frequency, therefore, reducing the error and improving usability of the proposed solution.

6.4. *Running example*

Continuing the example to show the working of proposed solution, Table 4 presents the conversion of closed rare itemsets into context-based event patterns, along with their temporal sequencing and calculating their degree of irregularity. Considering OBM_{D_1}, the rare itemset $\{B,F\}$ was found in E_5, which is assigned 5 as the sequence identifier and belongs to $ET4$ event type. As this itemset was not found in any other entry, it was excluded from the process. The same applies to the $\{C, D, F\}$, $\{A, E, F\}$ and $\{A, D, E\}$ closed rare itemsets as their context-based patterns only consist of single unique event type.

On the other hand, the rare itemset $\{C,D\}$ produced $ET1_1 \rightarrow ET5_6$, $\{A,E\}$ produced $ET2_2 \rightarrow ET3_4$ and $\{E,F\}$ produced $ET1_3 \rightarrow ET3_4$ patterns. It is evident that the proposed solution successfully found all connections, which were present amongst the given event entries. The only event which is not part of any pattern is $ET4_5$ as the objects of this entry $\{B,F\}$ do not correlate with the objects of any other entry. Furthermore, there are no repetitive patterns due to the simplistic nature of the example dataset. Every pattern represents an activity that only occurred once in the dataset, therefore, has the same degree of irregularity of one. From the forensic investigation's perspective, this means all activities are rare and should be probed with the same importance.

Algorithm 2 Counting the number of times a pattern occurred in the event log dataset.

```
Input: Object-based model of event log entries $OBM_D$
Input: Event type pattern $p_{ET}$
Output: A whole number $PatternCounter$ to show how many times a certain pattern of event types occurred in $OBM_D$
 1: function COUNT-PATTERN-OCCURRENCE($p_{ET}$, $OBM_D$)
 2:     Initialise $i \leftarrow 0$
 3:     Initialise $PatternCounter \leftarrow 0$
 4:     while $i < |OBM_D|$ do
 5:         Initialise $j \leftarrow 0$
 6:         Initialise $TempCounter \leftarrow 0$
 7:         while $j < |p_{ET}|$ and $i < |OBM_D|$ do
 8:             if $OBM_{D_i.T} = p_{ET_j}$ then
 9:                 $j \leftarrow j + 1$
10:                 $TempCounter \leftarrow TempCounter + 1$
11:             end if
12:             $i \leftarrow i + 1$
13:         end while
14:         if $TempCounter = |p_{ET}|$ then
15:             $PatternCounter \leftarrow PatternCounter + 1$.
16:         end if
17:     end while
18:     return $PatternCounter$
19: end function
```

#	Closed Rare itemsets	Context-based patterns	Temporal sequencing	Degree of irregularity
1	B, F	$ET4_5$	×	-
2	C, D	$ET1_1$, $ET5_6$	$ET1_1 \rightarrow ET5_6$	1
3	C, D, F	$ET5_6$	×	-
4	A, E	$ET2_2$, $ET3_4$	$ET2_2 \rightarrow ET3_4$	1
5	A, E, F	$ET3_4$	×	-
6	A, D, E	$ET2_2$	×	-
7	E, F	$ET1_3$, $ET3_4$	$ET1_3 \rightarrow ET3_4$	1

Table 4: Using closed rare itemsets to identify irregular activities, along with their degree/level of irregularity.

7. Empirical analysis

This section presents an empirical analysis of the developed solution using the event logs acquired from 15 live machines. The term 'live' means that all machines used for forensic analysis operate in the real-world. The reason behind analysing multiple machines is to verify the feasibility and usability of the solution under different circumstances. This evaluation process demonstrates the accuracy of the solution based on its ability to effectively uncover irregular activities within the selected machines, along with the efficiency and productivity improvements over the manual log analysis approach that requires effort, time, and expert knowledge. This section starts by describing the evaluation methodology. After that, it provides results and their discussion. In the end, a practical example is shown of an irregular activity extracted from one of the evaluation dataset.

7.1. Evaluation Methodology

Following explains the evaluation process that has been used to analyse the proposed solution on live computers:

1. *Data acquisition* – To obtain evaluation data, 15 computers are selected and investigated that were part of a university network domain, before being moved in to quarantine due system administrators suspecting that they have been involved in illegitimate activities, such failed login attempts, altering system configurations, privilege escalation attack to gain administrative access to files and applications, unauthorised modification of registry values, and creation of illegal user accounts. The computers contain regular Microsoft Windows operating

system (a mixture of both versions 7 and 10) and were used by multiple types of users, such as students, staff, and administrators. Each computer is configured to retain events for past 90 days. The usage of computers depends on the user's role and requirements (E.g., web surfing, communication and file sharing, accessing shared resources, research and development, database management, security compliance, and maintenance). Each type of role has a separate set of security policies and permissions, which are known and implemented by the administrators. The interpretation of the security policies provided the ground truth to compare the accuracy of the developed solution.

2. *Automated identification of irregular activities* – The next step is to retrieve event logs in their default (.evtx) format from the machines. After that, each event log is analysed by the developed tool to identify irregular activities;

3. *Expert identification of irregular activities* – Using the same event logs, irregular activities are collectively identified by two independent forensic computer analysts (working at the university), without looking at the results from the automated solution. Both analysts are expert investigators and examine system event logs as part of their job (when needed). The reason behind using two expert opinions is to gain impartial evaluation results. Several online resources[2,3] for detecting suspicious event patterns have also been utilised by the experts to validate and concur the results.

4. *Performance* – A comparison is conducted between the irregular activities, which were identified manually (by the two experts) and automatically (by the developed solution), to test the performance of our approach. The comparison is based on the number of correct, incorrect, missing, and regular activities:

   (a) An irregular activity identified by the developed solution is considered correct (or a true positive) if the same activity was flagged as irregular manually.

   (b) An irregular activity is considered incorrect (or a false positive), if it was found by the developed solution, but the manual analysis found it to be a regular event pattern.

   (c) An irregular activity is considered missing (or a false negative) if it was manually identified by the experts, but the developed solution failed to discover it as irregular.

   (d) An activity is deemed regular (routine or a true negative) if it was neither identified as irregular by the experts nor the developed solution.

   After collecting these values, the following performance metrics are calculated:

$$Classification\,Accuracy = \frac{correct + regular}{correct + incorrect + missing + regular} \times 100 \tag{1}$$

$$Recall = \frac{correct}{correct + missing} \times 100 \tag{2}$$

$$Specificity = \frac{incorrect}{incorrect + regular} \times 100 \tag{3}$$

$$Precision = \frac{correct}{correct + incorrect} \times 100 \tag{4}$$

$$F - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \times 100 \tag{5}$$

[2]https://www.malwarearchaeology.com/cheat-sheets/

[3]https://apps.nsa.gov/iaarchive/library/ia-guidance/security-configuration/applications/spotting-the-adversary-with-windows-event-log-monitoring.cfm

7.2. Results and Discussion

The empirical analysis of the developed solution is shown in Table 5. The analysis was performed on a 64-bit Microsoft Windows 10 machine with Intel Core i7-8700 3.20GHz CPU and 16GB RAM. The results are presented from each phase of processing, such as the number of rare itemsets, closed rare itemsets, irregular activities, time is taken and accuracy. The solution has demonstrated reasonable accuracy, along with the benefit of workload reduction over the manual analysis approach. Notice that there is an element of ambiguity in what constitutes as an irregular activity in some cases. A single activity can be performed using different methods to achieve the same goal and each method of execution of the activity might generate different events or involve the same events in a different order. It is likely that the experts do not know every order and combination of events corresponding to every system activity. Hence, there is potential for a degree of uncertainty or error in finding the accuracy.

The classification accuracy is a proportion of correct predictions to the total number of predictions (Equation 1) and it ranges between 75.68% and 91.67% for the evaluation data. Dataset-3 and dataset-9 showed the lowest and dataset-10 showed the highest accuracy. In terms of total processing time, 522 entries of dataset-3 were processed in five minutes and 14 seconds, whereas, dataset-13 containing 69,856 entries was analysed in 41 minutes and 25 seconds. The developed solution saved a significant amount of time and effort, considering that most of the irregular activities were successfully recognised without needing any kind of assistance or input from the human investigator.

Recall, Specificity, and Precision are important metrics to fully evaluate and validate the developed solution. Recall shows the ratio of actual positives that were correctly identified by the solution (Equation 2) and it ranges between 66.67% and 100%. Specificity presents the ratio of true negatives (or regular activities) that was correctly excluded by the solution from the results (Equation 3). The range of Specificity is between 68.75% and 100%, which demonstrates the solution's ability to distinguish between rare and frequent activities. Precision is the ratio of positive instances that were actually correct (Equation 4) and it ranges between 66.67% and 100%. It should also be noted here that the cost of false positives and false negatives are not the same in our case. If there is a high number of false positive results (i.e., false alarm to non-existent irregular activities), it would affect the investigator's performance as they would be spending more time investigating and removing benign event patterns. Conversely, if there is a high number of false negatives, it indicates a relatively more critical issue that the solution has poor performance and capability due to failing to recognise existing irregular (and possibly malicious) activities. Therefore, it is necessary to take both false positive and false negative results into consideration and determine the proportion of those correctly identified as actual irregular activities in the dataset. This is depicted by F-Score metric, calculated as the weighted average of precision and recall (Equation 5), which ranges between 71.43% and 85.71% for the 15 datasets. These metrics demonstrate better classification performance, meaning that our solution can successfully find most of the relevant, irregular event patterns in a given dataset.

A significant aspect of the proposed solution is the reduction in the number of rare itemsets without losing any information. A large segment of itemsets are removed from each dataset by finding and eliminating the non-closed rare itemsets. For example, more than 99% of the itemsets were discarded by the solution generated from dataset-1 and the remaining ones presented seven rare activities. Similarly, 82% of the non-closed rare itemsets were removed from dataset-12 and the rest exhibited 17 irregular activities. Converting the closed rare itemsets into context-based patterns (as described in Section 6.1) generates a precise and the most relevant set of connected event entries. This enables the investigator to easily identify and analyse the suspicious activities performed on the machine.

It is worth noting that some of the identified irregular activities in the datasets show more than one occurrence. For example, the developed solution identified three irregular activities in dataset-7, each occurring between three to five times. Similarly, the solution identified three irregular activities in dataset-6, where their occurrence ranges from two to five times. Although the identified activities are not perfectly rare (i.e., more than one occurrence of the event pattern in the dataset), they still are relatively rare as compared to other activities and might contain important pieces of information relevant to the investigation.

Another important aspect of the solution is the (small) amount of entries involved in the identified activities. For example, only 1.4% of the total entries of dataset-9 are involved in representing its irregular activities, i.e., the investigator only needs to inspect a small number of entries to determine and analyse the discovered irregular activities. Similarly, the investigator can ignore most of the entries in dataset-8 and acquire a comprehensive view of the suspicious activities. This signifies the efficiency of the developed solution and can help in reducing the time required to undertake a forensic investigation of event logs.

Log	No. of event entries	Rare itemsets	Closed rare itemsets	No. of irregular activities	Pattern occurrence range	Entries involved in activities	Time taken (mm:ss)	Accuracy %	Recall %	Specificity %	Precision %	F-score %
1	5,028	57,746	58	7	2–4	7.52	18:47	82.61	77.78	85.71	77.78	77.78
2	8,254	150,075	835	3	1	5.95	25:49	90.91	75	94.4	75	75
3	522	13,830	169	2	3–4	10.34	05:14	85.71	100	80	66.67	80
4	1,123	15,606	160	8	1–5	19.06	06:38	76.47	72.73	83.33	88.89	80
5	1,080	17,148	187	4	2–5	8.8	07:11	78.57	66.67	87.5	80	72.73
6	1,692	11,720	122	3	2–5	7.68	05:24	85.71	75	90	75	75
7	1,715	21,569	206	3	3–5	8.1	10:10	90	75	100	100	85.71
8	12,976	29,063	148	3	1–2	4.61	23:34	91.67	75	95	75	75
9	1,073	21,731	132	2	3–3	1.4	09:50	88.89	66.67	100	100	80
10	1,992	77,065	261	12	1–5	21.99	08:55	79.17	80	77.78	85.71	82.76
11	827	19,341	118	8	1–5	30.59	08:30	81.25	88.89	71.43	80	84.21
12	5,310	16,992	206	17	1–5	25.86	11:08	75.68	80.95	68.75	77.27	79.07
13	69,856	31,181	407	5	1–3	16.08	41:25	83.33	71.43	88.24	71.43	71.43
14	19,572	1,578	102	5	1–5	13.04	19:28	90	83.3	92.86	83.33	83.33
15	15,642	54,945	275	7	1–5	6.44	24:23	84.62	77.78	88.26	77.78	77.78

Table 5: Empirical analysis results gathered from each stage of the developed solution for identifying irregular activities in 15 (real-world) event log datasets.

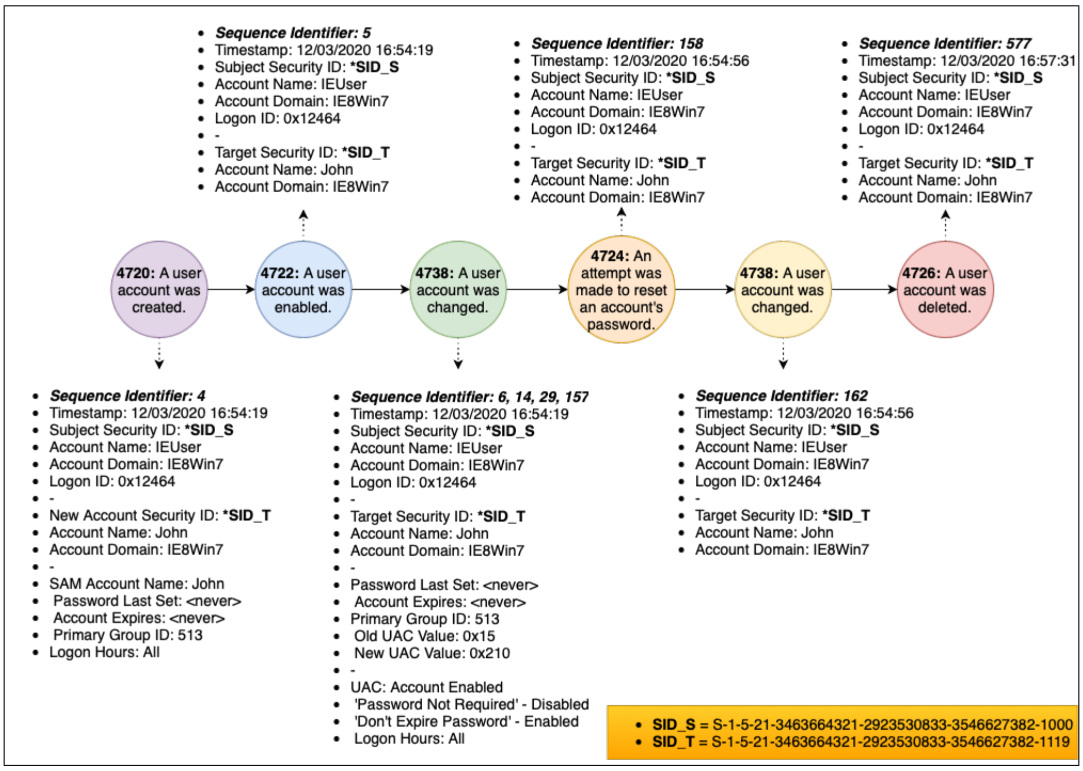
7.3. Example of an irregular activity

A complete timeline demonstrating an irregular activity found in live dataset-13 is presented in Figure 4. According to the results, this activity occurred only once in the dataset, involves five unique event types (IDs of: 4720, 4722, 4738, 4724, and 4726), and spans over nine entries. This activity provides an artifact of a certain user account deletion, which is considered a useful discovery during forensic analysis. A research study claims that one of the common ways of concealing malicious activities is to create a user account, launch attack(s) through the new account, and then delete it immediately to remove all data and traces (Al-Saleh & Al-Shamaileh, 2017). The extracted activity is described in the following:

1. The first entry in the activity has *4720* event type, with a sequence identifier of 4. This event shows that a user account with the name of `IEUser` and subject security identifier (SSID) as `S-1-5-21-3463664321-2923530833` `-3546627382-1000` created a new account on domain `IE8Win7`. The new account, named `John`, is assigned `S-1-5-21-3463664321-2923530833-3546627382-1119` target security identifier (TSID) and resides in the same domain. Both SSID and TSID have unique values of variable length to identify a trustee and are issued by an authority.

2. The next entry has *4722* event type, with a sequence identifier as five. This entry shows that the account was enabled, right after it was created. According to Microsoft security monitoring recommendation[4], this type of event should always be monitored for high-value accounts, E.g., administrator, database administrator and domain controller.

[4]https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4722


4720: A user account was created.
4722: A user account was enabled.
4738: A user account was changed.
4724: An attempt was made to reset an account's password.
4738: A user account was changed.
4726: A user account was deleted.

- ***Sequence Identifier: 4***
- Timestamp: 12/03/2020 16:54:19
- Subject Security ID: ***SID_S**
- Account Name: IEUser
- Account Domain: IE8Win7
- Logon ID: 0x12464
- -
- New Account Security ID: ***SID_T**
- Account Name: John
- Account Domain: IE8Win7
- -
- SAM Account Name: John
- Password Last Set: <never>
- Account Expires: <never>
- Primary Group ID: 513
- Logon Hours: All

- ***Sequence Identifier: 5***
- Timestamp: 12/03/2020 16:54:19
- Subject Security ID: ***SID_S**
- Account Name: IEUser
- Account Domain: IE8Win7
- Logon ID: 0x12464
- -
- Target Security ID: ***SID_T**
- Account Name: John
- Account Domain: IE8Win7

- ***Sequence Identifier: 6, 14, 29, 157***
- Timestamp: 12/03/2020 16:54:19
- Subject Security ID: ***SID_S**
- Account Name: IEUser
- Account Domain: IE8Win7
- Logon ID: 0x12464
- -
- Target Security ID: ***SID_T**
- Account Name: John
- Account Domain: IE8Win7
- -
- Password Last Set: <never>
- Account Expires: <never>
- Primary Group ID: 513
- Old UAC Value: 0x15
- New UAC Value: 0x210
- -
- UAC: Account Enabled
- 'Password Not Required' - Disabled
- 'Don't Expire Password' - Enabled
- Logon Hours: All

- ***Sequence Identifier: 158***
- Timestamp: 12/03/2020 16:54:56
- Subject Security ID: ***SID_S**
- Account Name: IEUser
- Account Domain: IE8Win7
- Logon ID: 0x12464
- -
- Target Security ID: ***SID_T**
- Account Name: John
- Account Domain: IE8Win7

- ***Sequence Identifier: 162***
- Timestamp: 12/03/2020 16:54:56
- Subject Security ID: ***SID_S**
- Account Name: IEUser
- Account Domain: IE8Win7
- Logon ID: 0x12464
- -
- Target Security ID: ***SID_T**
- Account Name: John
- Account Domain: IE8Win7

- ***Sequence Identifier: 577***
- Timestamp: 12/03/2020 16:57:31
- Subject Security ID: ***SID_S**
- Account Name: IEUser
- Account Domain: IE8Win7
- Logon ID: 0x12464
- -
- Target Security ID: ***SID_T**
- Account Name: John
- Account Domain: IE8Win7

- **SID_S** = S-1-5-21-3463664321-2923530833-3546627382-1000
- **SID_T** = S-1-5-21-3463664321-2923530833-3546627382-1119

Figure 4: Example of an irregular activity extracted by the proposed solution from dataset-13 event log. Note that both labels SID_S and SID_T are defined at the right-bottom of the figure.

3. The next four entries belong to *4738* event type, having 6, 14, 29, and 157 sequence identifiers. This type of event is triggered every time an object is changed in the user account configuration. An individual event will be recorded for each change, based on the Microsoft documentation[5]. The proposed solution extracted four entries with the same event type and the same SSID and TSID to reflect four different user account changes. Although these event entries seem trivial, they have revealed important information for forensic analysis regarding the user account `John`:

   - The account is set to never expire, so it will remain active until manually disabled or removed; and
   - The account password will never expire, indicating only the creator of this account will be able to use it.

4. A combination of the next two entries of event type *4724* and *4738* shows that the password, which was set at the time of account creation, is changed.

5. The final event entry belongs to the *4726* event type and has 577 sequence identifier. This entry, having the same SSID and TSID as all previous entries, was recorded with a time gap and informs that the user account `John` was removed from the underlying machine. It is of paramount importance to notice the time span between the first and last entry of this entire activity. All entries were generated between *16:54:19* and *16:57:31* on the same date *12/03/2020*. This informs that the user account was created and removed within three minutes and 12 seconds.

As creating and removing an account within such a time span is not common in a university machine, and this administrative process only occurred once in the dataset, this activity is correctly classified as a rare activity. Therefore,

[5]https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4738

it would be beneficial to investigate the underlying machine in-depth (using other available resources, E.g., Registry, System Resource Utilisation Monitor, and Network activity) and determine if `John` user account has performed any malicious activities. This demonstrates the efficiency and effectiveness of the developed solution in discovering a record of anomalous activity using automated forensic analysis of the event logs.

7.4. Limitations of our approach

1. Depending on the machine under examination, it is likely that the developed solution might produce false positives, which will need to be eliminated from processing by the human investigator. For instance, when a user accidentally enters a wrong username/password during login, it could be identified as a rare activity.

2. Execution time grows rapidly for larger datasets. This requires further evaluation to determine the solution's ability and performance to operate in large-scale environments. However, comparing it with the manual log analysis, the presented automated tool is still useful, saving valuable time and effort during the investigation.

8. Conclusion and Future work

This paper presents a novel automated technique to identify irregular activities from system event logs in an automated manner. The main purpose of this solution is to enable forensic investigators to conduct efficient analysis of previously unseen machines, without spending a significant amount of time and effort in examining their event entries. The presented technique was developed and tested using 15 event log datasets. First, it creates an object-based model to represent event log entries and applies rare itemset mining techniques using 1% and 50% support thresholds. The itemsets are then formulated into compressed representations to reduce processing time. After that, all itemsets are converted into respective context-based patterns and ordered based on a temporal metric to denote the irregular activities. The solution also allows for differentiation among patterns based on their degree of irregularity. Although the event logs usually contain a large number of entries, both relevant and irrelevant to the forensic investigation, the proposed solution demonstrated that it can successfully determine the key irregular activities with reasonable accuracy and performance in practical environments.

Future work – As mentioned in the literature review (Section 2), process mining is another potential solution for event log analysis to identify trends, patterns, and details of how a (regular or irregular) process unfolds. In the future, we will explore the use and feasibility of process mining algorithms to discover interesting system activities and identify irregularities or deviations. We can then evaluate how our proposed approach (closed rare itemset mining) compares with process mining in terms of performance during forensics investigations. We also plan to augment the developed solution to analyse multiple and possibly heterogeneous event logs from different sources, simultaneously. This collective analysis will assist the investigator to detect distributed irregular and anomalous activities across the entire network. Another future goal is to devise an automated mechanism to determine support values for each event log dataset separately, in order to improve the solution's autonomy. To increase the usability and applicability of the tool, we also plan to develop a new component in the proposed solution that is capable of parsing and generating an object-based model from different event log formats (I.e., structured, semi-structured, and unstructured).

References

Ahmed, M. (2016). *Detecting rare and collective anomalies in network traffic data using summarization*. Ph.D. thesis Ph. d. theses, UNSW Australia.

Al-Saleh, M. I., & Al-Shamaileh, M. J. (2017). Forensic artefacts associated with intentionally deleted user accounts. *International Journal of Electronic Security and Digital Forensics*, *9*, 167–179.

Bahaweres, R. B., Trawally, J., Hermadi, I., & Suroso, A. I. (2021). Forensic audit using process mining to detect fraud. In *Journal of Physics: Conference Series* (p. 012013). IOP Publishing volume 1779.

Bakariya, B., & Thakur, G. (2016). Mining rare itemsets from weblog data. *National Academy Science Letters*, *39*, 359–363.

Batal, I., Valizadegan, H., Cooper, G. F., & Hauskrecht, M. (2013). A temporal pattern mining approach for classifying electronic health record data. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *4*, 1–22.

Borah, A., & Nath, B. (2019). Rare pattern mining: challenges and future perspectives. *Complex & Intelligent Systems*, *5*, 1–23.

Chabot, Y., Bertaux, A., Kechadi, T., & Nicolle, C. (2015). Event reconstruction: A state of the art. In *Handbook of Research on Digital Crime, Cyberspace Security, and Information Assurance* (pp. 231–245). IGI Global.

Chiu, T., & Jans, M. (2019). Process mining of event logs: A case study evaluating internal control effectiveness. *Accounting Horizons*, *33*, 141–156.

Darrab, S., Broneske, D., & Saake, G. (2021). Modern applications and challenges for rare itemset mining. *International Journal of Machine Learning and Computing*, *11*.

De Leoni, M., van der Aalst, W. M., & Dees, M. (2016). A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. *Information Systems*, *56*, 235–257.

Dijkman, R., & Wilbik, A. (2017). Linguistic summarization of event logs–a practical approach. *Information systems*, *67*, 114–125.

Djenouri, Y., Belhadi, A., & Fournier-Viger, P. (2018). Extracting useful knowledge from event logs: a frequent itemset mining approach. *Knowledge-Based Systems*, *139*, 132–148.

Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1285–1298).

Feremans, L., Vercruyssen, V., Cule, B., Meert, W., & Goethals, B. (2019a). Pattern-based anomaly detection in mixed-type time series. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 240–256). Springer.

Feremans, L., Vercruyssen, V., Meert, W., Cule, B., & Goethals, B. (2019b). A framework for pattern mining and anomaly detection in multi-dimensional time series and event logs. In *International Workshop on New Frontiers in Mining Complex Patterns* (pp. 3–20). Springer.

Fournier-Viger, P., Lin, J. C.-W., Vo, B., Chi, T. T., Zhang, J., & Le, H. B. (2017). A survey of itemset mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *7*, e1207.

Garg, R., & Gulia, P. (2015). Comparative study of frequent itemset mining algorithms apriori and fp growth. *International Journal of Computer Applications*, *126*.

Hendricks, R. M. (2019). Process mining of incoming patients with sepsis. *Online journal of public health informatics*, *11*.

Ibrahim, N. M., Al-Nemrat, A., Jahankhani, H., & Bashroush, R. (2011). Sufficiency of windows event log as evidence in digital forensics. In *Global Security, Safety and Sustainability & e-Democracy* (pp. 253–262). Springer.

Islam, R. U., Hossain, M. S., & Andersson, K. (2018). A novel anomaly detection algorithm for sensor data under uncertainty. *Soft Computing*, *22*, 1623–1639.

James, J. I., & Gladyshev, P. (2013). Challenges with automation in digital forensic investigations. *arXiv preprint arXiv:1303.4498*, .

Jans, M., Alles, M. G., & Vasarhelyi, M. A. (2014). A field study on the use of process mining of event logs as an analytical procedure in auditing. *The Accounting Review*, *89*, 1751–1773.

Kalenkova, A. A., van der Aalst, W. M., Lomazova, I. A., & Rubin, V. A. (2017). Process mining using bpmn: relating event logs and process models. *Software & Systems Modeling*, *16*, 1019–1048.

Khan, S., & Parkinson, S. (2017). Causal connections mining within security event logs. In *Proceedings of the Knowledge Capture Conference* (pp. 1–4).

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.

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

Khatuya, S., Ganguly, N., Basak, J., Bharde, M., & Mitra, B. (2018). Adele: Anomaly detection from event log empiricism. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications* (pp. 2114–2122). IEEE.

Koh, Y. S., & Rountree, N. (2005). Finding sporadic rules using apriori-inverse. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 97–106). Springer.

Landauer, M., Wurzenberger, M., Skopik, F., Settanni, G., & Filzmoser, P. (2018). Dynamic log file analysis: An unsupervised cluster evolution approach for anomaly detection. *computers & security*, *79*, 94–116.

Li, T., Jiang, Y., Zeng, C., Xia, B., Liu, Z., Zhou, W., Zhu, X., Wang, W., Zhang, L., Wu, J. et al. (2017). Flap: An end-to-end event log analysis platform for system management. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1547–1556).

Lillis, D., Becker, B., O'Sullivan, T., & Scanlon, M. (2016). Current challenges and future research areas for digital forensic investigation. *arXiv preprint arXiv:1604.03850*, .

Martin, T., Francoeur, G., & Valtchev, P. (2020). Ciclad: A fast and memory-efficient closed itemset miner for streams. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 1810–1818).

de Murillas, E. G. L., Reijers, H., & van der Aalst, W. (2020). Case notion discovery and recommendation: automated event log building on databases. *Knowledge and Information Systems*, *62*, 2539–2575.

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.

Rahman, A. (2019). *Rare sequential pattern mining of critical infrastructure control logs for anomaly detection*. Ph.D. thesis Queensland University of Technology.

Rahman, A., Xu, Y., Radke, K., & Foo, E. (2016). Finding anomalies in scada logs using rare sequential pattern mining. In *International Conference on Network and System Security* (pp. 499–506). Springer.

Ranjan, R., & Sharma, A. (2019). Evaluation of frequent itemset mining platforms using apriori and fp-growth algorithm. *International Journal of Information Systems & Management Science*, *2*.

Romero, C., Romero, J. R., Luna, J. M., & Ventura, S. (2010). Mining rare association rules from e-learning data. In *Educational Data Mining 2010*. ERIC.

Sadhasivam, K. S., & Angamuthu, T. (2011). Mining rare itemset with automated support thresholds. *Journal of Computer Science*, *7*, 394.

Sanjappa, S., & Ahmed, M. (2017). Analysis of logs by using logstash. In *Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications* (pp. 579–585). Springer.

dos Santos Garcia, C., Meincheim, A., Junior, E. R. F., Dallagassa, M. R., Sato, D. M. V., Carvalho, D. R., Santos, E. A. P., & Scalabrin, E. E. (2019). Process mining techniques and applications–a systematic mapping study. *Expert Systems with Applications*, *133*, 260–295.

Selvi, C. S. K., Malliga, S., & Kogilavani, S. V. (2016). Automated support thresholds for rule mining. *International Journal of Business Intelligence and Data Mining*, *11*, 151–170.

Siddiqui, M. A., Fern, A., Dietterich, T. G., & Das, S. (2016). Finite sample complexity of rare pattern anomaly detection. In *UAI*.

Studiawan, H., Sohel, F., & Payne, C. (2019). A survey on forensic investigation of operating system logs. *Digital Investigation*, *29*, 1–20.

Studiawan, H., Sohel, F., & Payne, C. (2020). Automatic event log abstraction to support forensic investigation. In *Proceedings of the Australasian Computer Science Week Multiconference* (pp. 1–9).

Szathmary, L., Napoli, A., & Valtchev, P. (2007). Towards rare itemset mining. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)* (pp. 305–312). IEEE volume 1.

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

Tsang, S., Koh, Y. S., & Dobbie, G. (2011). Rp-tree: rare pattern tree mining. In *International conference on data warehousing and knowledge discovery* (pp. 277–288). Springer.

Vaarandi, R., Kont, M., & Pihelgas, M. (2016). Event log analysis with the logcluster tool. In *MILCOM 2016-2016 IEEE Military Communications Conference* (pp. 982–987). IEEE.

Van Dongen, B. F., de Medeiros, A. K. A., Verbeek, H., Weijters, A., & van Der Aalst, W. M. (2005). The prom framework: A new era in process mining tool support. In *International conference on application and theory of petri nets* (pp. 444–454). Springer.

Wang, L., Du, Y., & Qi, L. (2019). Efficient deviation detection between a process model and event logs. *IEEE/CAA Journal of Automatica Sinica*, *6*, 1352–1364.

Wu, C.-W., Lin, Y.-F., Yu, P. S., & Tseng, V. S. (2013). Mining high utility episodes in complex event sequences. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 536–544).

Zaki, F. A. M., & Zulkurnain, N. F. (2018). Rare: Mining colossal closed itemset in high dimensional data. *Knowledge-Based Systems*, *161*, 1–11.

Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., & Lyu, M. R. (2019). Tools and benchmarks for automated log parsing. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (pp. 121–130). IEEE.