DRAFT Benchmarking a Parameterised Approach to Generating and Detecting Event Chains in Security Dataset

Saad Khan[a,*], Simon Parkinson[a], Monika Roopak[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-based datasets are fundamental in cyber security applications, forming an essential part of defensive applications. One fundamental use case focused on in this paper is the detection of event chains, which contain information relating to an activity that spans many individual events, and an understanding can only be acquired once they are identified and chained together. As such, researchers use event datasets when developing and testing new techniques. Their testing often includes a combination of real-world and synthetic analysis. However, due to the security-sensitive nature of the datasets, they are often not shared, making it challenging to establish comparative capability and performance between techniques. In this article, we present a novel parameterised event-based generation technique, capable of producing synthetic datasets as well as an approach for adding synthetic event chains into real-world datasets for detection. We then provide a benchmark in event chain detection through the use of clustering techniques. Our benchmarking demonstrated the suitability of DBSCAN to detect event chains, achieving greater than a 0.95 Adjusted Rand Index score on most of the generated 12K event log files.

Keywords: Event Log Generation, Event Analysis, Clustering, DBSCAN

1. Introduction

Throughout IT systems, event logging mechanisms store information on system activity that may be required during future analysis to understand system use and behaviour. The information could be helpful for debugging activities, understanding system usage and designing and improving system's security. Security events include information, warnings, and errors generated by security controls (e.g., authentication and access controls) and security services (e.g., antivirus). This information provides essential information for reviewing system security to understand issues arising, but also to review post-incident to learn how the incident occurred to enable the prevention of future occurrences.

Security events are generated and stored on a device level. In the majority of computer networks, individual devices are often referred to as end points. The scale of their generation in terms of frequency and diversity, manually inspecting event logs on endpoint devices would not be feasible. It would require large amounts of manual resources to inspect, and the scale of the events is often too large for a human to comprehend. It is for this reason that security monitoring software solutions exist, which have the capability of processing event logs from endpoints. These solutions are often categorised as being Security Information, and Event Management (SIEM) (Parkinson et al., 2018), Security Operation Centre (SOC), or Intrusion Detection solutions, among others. These solutions typically use predefined knowledge (Khan & Parkinson, 2017, 2018, 2019) or statistics to establish known sequences of events that are troublesome or characteristics

*Corresponding author
Email addresses: saad.khan@hud.ac.uk (Saad Khan), s.parkinson@hud.ac.uk (Simon Parkinson), m.roopak@hud.ac.uk (Monika Roopak)

of the data that are unusual or anomalous (Parkinson & Khan, 2018). It is also the case that events are used in different security auditing processes, such as monitoring and refining access controls (Khan & Parkinson, 2022). The functionality of this type typically requires a combination of supervised and unsupervised learning, where software is trained to recognise patterns of interest, such as a known or anomalous sequence of events. A linked sequence of events, or an event chain, describes a series of events that are linked to the same activity, which in the context of this paper could be an attack or a change in the security configuration.

Event chains are often established by security researchers and practitioners as they are a useful template that can be used to detect known and defined system use and behaviour, which can be thought of as an attack trace. These event chains are typically manually constructed through post-incident analysis and an understanding of system security controls and logging behaviour. The construction of event chains is a reactive and largely manual process. This is because, to gain an understanding of a security incident or a cyber attack, researchers need to determine which event chain was triggered in a precise manner. The discovered event chain is also utilised by a pattern-matching software to detect any future instances. The area of processing mining involves the use of algorithms to identify linked activities within an event log (Suriadi et al., 2017). Whether through manual or automated extraction, the modelling and relationships of event contents in an object-centric manner have become an integral part of maintaining system security (Ghahfarokhi et al., 2021).

Developing techniques that are capable of automatically learning event chains is an ongoing research area. However, determining the capability of these techniques depends on the availability of appropriate testing dataset. Data extracted from live systems does not contain ground-truth information, while the synthetically generated dataset lack sufficient variation and comprehensiveness to represent real-world scenario Alshaikh et al. (2022). In addition to the use of event chains in security monitoring and defence, the ability to detect individual anomalous events is also beneficial. For example, there are some security events that can occur on a system that might result in the generation of a singular event, such as user authentication. For example, a user logging into a different PC for the first time, at an unusual time, or even an incorrect authentication attempt, could all be seen as anomalous events.

The development of SIEM and SOC solutions is dependent on data that can be used for training and testing. Data has a significant role in their development. Without realistic and representative data, the capabilities of the system will be inherently limited. Despite the importance of data, there is an absence of a reference dataset or approach to generate realistic datasets for system development and benchmarking. Datasets that are available are often imbalanced and mostly focused on anomaly detection, not event chain mining techniques. Further, the inability to have known ground truth and flexibility over completeness and heterogeneity restricts developing systems capable of handling the diverse characteristics of data seen in real-world systems. This constrains the development and comparison of techniques processing event-based data sources.

In this paper, a systematic and parameterised approach is presented for generating event-based datasets. The technique provides the user with flexibility in generating event datasets, considering aspects, such as the size and characteristics of event chains. The technique is repeatable and enables researchers and developers to benchmark different techniques. In addition to presenting the parameterised technique, we also demonstrate how it can be used for both the detection of anomalies and event chains. The paper is structured as follows:

In Section 2, a brief review of relevant literature is provided, serving as a key motivator of the presented work. Following on, Section 3 presents a background on event logging, focusing on how their object-based construction is essential to event chain extraction. In Section 4, the parameterised dataset generator is presented. In Section 5, a brief introduction to the different clustering algorithms used in this benchmarking is included, before presenting preliminary results in Section 7. The outcome of the preliminary analysis motivates a full analysis in Section 8 using DBSCAN. Finally, in Section 9 a conclusion and future work are provided.

2. Related Work

Due to significant growth in computing infrastructure, organisations commonly have to consistently and proactively collect and analyse large volumes of event log data for various security purposes (He et al.,

2021), such as understanding user, system, and application-specific activities, detecting anomalies to prevent security breaches, identifying the root cause of system disruptions, etc. Analysing events also provides visibility of the working of all underlying system components in a data-rich environment which is always a crucial priority for organisations undertaking monitoring tasks. Existing research demonstrates that there are many techniques that can be utilised to extract valuable information from the event logs, such as frequent or rare pattern mining (Li, 2015), deep learning (Chen et al., 2021), and clustering (Delias et al., 2015).

It is evident that a substantial part of existing research focuses on utilising clustering techniques for event log analysis. Data clustering (Xu & Wunsch, 2005) is an unsupervised process of grouping items or event entries (Vaarandi, 2003) of an unlabelled dataset, such that items in each cluster are similar to each other with respect to their features and attributes and as dissimilar as possible to items from other clusters. This process creates one or more segregated groups of entries, where each entry is more related to other entries of the same group than those of the other groups. Clustering algorithms can be broadly classified into Partitional and Hierarchical schemes (Rai et al., 2011). The Partitional clustering schemes are parametric where algorithms, such as K-means, are informed by the number of clusters to create within the input dataset. The Hierarchical clustering schemes are non-parametric which means algorithms, such MeanShift, decide how many clusters to create based on certain intrinsic metrics. In this paper, our focus is to utilise Hierarchical clustering schemes as it is not possible to have prior knowledge of clusters present in an event dataset. Using clustering to find relationships among events provides valuable insight into the event logs as it will discover related events based on their high degree of common objects, therefore, grouping events that were likely triggered by the same user, application, or system activity. The clustering algorithms we have selected for benchmarking purposes are explained in Section 5.

A suitable dataset is always required to evaluate the capability of an event log analysis technique fully. The dataset should be entirely known in that its contents are fully understood. In this section, we present related event log datasets that are employed in recent research works for various applications. Details regarding each dataset are summarised in Table 1. The MIMICEL (Wei et al., 2022) dataset consists of patients' activities and their complete journey in the emergency department. The dataset was created to improve the efficiency of the department and contains events from 425,087 patient cases. An application named BELA (Blockchain Event Log App) (Alzhrani et al., 2023) has been developed to collect the event logs from decentralised blockchain-based applications. The dataset is available in two forms; the raw version in which the data is encoded by blockchain and the other one is decoded version which presents a humanly readable version. An event log dataset (Špaček et al., 2022) is captured on a University network, from 8 real-time web servers, through the network and web-host monitoring of more than 800 websites for 7 days. The dataset consists of both custom features and IIS (Internet Information Services) default features. IIS logging information has also been used for the collection of events and the dataset formatting is made suitable for machine learning applications, such as anomaly detection, finding relationships between the server and network traffic events, etc. In another research study (Wu et al., 2022), the authors used 4 real-world event log datasets for building a predictive neural network-based model for cyber-attacks and threats. The first dataset, called Multi-log, has attack-related data from various sources, such as monitoring a system, its network traffic and files. Second, Web Application Firewall (WAF) log contains alerts triggered by attacks on a network collected by deployed web application firewall. Third, ARCS is a collection of event logs derived from an enterprise network of computers running Microsoft Windows. It comprises security events generated by the victim's machine during cyber attacks. The fourth dataset, CFDR Blue, has Reliability, Availability and Serviceability (RAS) log messages derived from the Blue Gene/P Intrepid system. Another work utilised event logs collected from real cyber-crime cases (Kara, 2021). It contains 3 three types of event logs: system, application and network. Another study used an event log dataset acquired from an Air traffic control system developed by a leading company (Cinque et al., 2020). The dataset contains events related to normative system behaviour, secure shell scan, denial of service attacks and flight rerouting scenarios, and is collected from 7 network nodes. In another work, authors used multiple event log datasets generated by information systems managing road traffic fines, hospital billing, building permit application process, Volvo's It incident and Sepsis, to evaluate their log sanitisation technique (Fahrenkrog-Petersen et al., 2023).

Many other event log datasets are also publicly available for use (Huang et al., 2020; Augusto et al., 2018; Studiawan et al., 2020) that come in different sizes and formats and were gathered from heterogeneous sources

Research study	Characteristics	Size
Wei et al. (2022)	Data of hospital's emergency department patients	7,887,229 instances
Alzhrani et al. (2023)	Decentralised blockchain-based application	101 files
MULTI log (private dataset)	Network and System	15 million entries
WAF (private dataset)	Network Traffic	272,000 entries
RCS Turcotte et al. (2019)	Host Event logs	23 million entries
CFDR Oliner & Stearley (2007)	RAS log messages	137,000 entries
Wu et al. (2022)	Network and system, host event logs, RAS log messages	2+ million entries
Špaček et al. (2022)	Network and host-based event logs	21 files
Kara (2021)	real cyber attack on a Microsoft system	Not available
Cinque et al. (2020)	Air traffic proprietary log data consisting of application and system logs	18 log files
De Leoni & Mannhardt (2015)	Event log of an information system managing road traffic fines	150,270 traces
Mannhardt (2017)	Event log was obtained from the financial modules of the ERP system of a regional hospital	177,751 traces
Buijs (2014)	Event logs of execution of building permit	1348 traces
Steeman (2013)	Logs of Volvo IT incident and problem management	1236 traces
Mannhardt et al. (2016)	Event log contains events of sepsis cases from a hospital	266 traces
Khan & Parkinson (2018, 2019)	20 end-point event logs from	ranging from 3-27K

Table 1: Characteristics of Event log datasets used in the existing research

for various purposes. However, these datasets are still largely unsuitable for benchmarking the performance of data mining techniques due to the absence of prior knowledge (i.e., ground truth and characteristics) about the data. If there is no knowledge of what activities an event log dataset contains, it is not possible to determine whether a data mining technique has worked correctly in identifying those activities. To resolve this, researchers have developed synthetic event log generators that provide several key advantages. First, every detail about what the generated event log file contains is already known to the user, thereby establishing the ground truth. Second, the user can generate realistic data, mostly for a specific application, based on a certain profile or pattern. Third, the user can generate event datasets of any size, providing sufficient data to fully gauge the performance of a newly developed or an existing data mining technique.

In one related research study, the authors developed a semi-synthetic technique to generate highly realistic network event sequence (NES) data that can be used to test network analysis software tools (Wurzenberger et al., 2016). The technique takes a small set of real network data as an input and applies log line clustering to group log entries and Markov chains to determine the correlation among groups. Through evaluation, the authors have also demonstrated the high degree of similarity between an original and generated log file. In another work, the authors present a tool to generate synthetic IoT sensor event logs, along with providing ground truth. Users can also configure the tool to, for example, add noise (erroneous events) to generate a more realistic dataset (Zisgen et al., 2022). A research study proposed a Synthetic Log Generator (SynLogGen) that can generate an event log dataset based on a business process profile defined as a Petri net in tabular form (Esgin & Karagoz, 2019). The profile includes process structure, noise threshold, priority, etc. The SynLogGen performs random executions (i.e., traversing) of the input Petri-net to produce sequences of activities. It also addresses the problem of unlabelled event log entries by adding identifiers to the process instances. However, the tool only supports Petri net input and is not flexible enough to accommodate different data mining applications. Follow on research extends SynLogGen and proposes a new solution called, Processes Logs Generator 2 (PLG2) (Burattin, 2015). It improves the previous random dataset generation approach by allowing the user to generate potentially infinite multi-perspective event streams to simulate realistic scenarios, such as controlling the amount and type of noise, integrating the notion of time in event data by changing event trigger rates, etc. Another similar work proposed PURPLE (PURPose guided Log gEneration) that performs user-guided executions of a given process profile (Burattin et al., 2022), instead of random that always outputs a different event log file every time, to incrementally generate a specific execution trace that satisfies any given desired purpose, such as to benchmark a data mining algorithm. While both PLG2 and PURPLE are flexible and can be tailored to user needs, they still have certain shortcomings in terms of only focusing on the control-flow aspects of the process and not including the process data (i.e., process context information) and constraints that can add new dimensions to the benchmark criteria. Another study extends existing research and presents SAMPLE (Grüger et al., 2023), which is a technique for a synthetic event log generator. In addition to generating multi-perspective event streams, SAMPLE also considers data perspective and its semantics. This helps in generating correct, meaningful and realistic

Permissions on an object were changed.
ID: log ID 4670
Subject:
Security ID: admin
Account Name: John
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 1: Example Microsoft event (ID 4670) detailing the change in security permissions

Apr 28 17:06:20 ip-172-31-11-241 sshd[1247]:
Invalid user admin from 216.19.2.8

Figure 2: Example Syslog event for an invalid user login

values for the process variables, consequently leading to better-quality synthetic data. However, SAMPLE has limitations in terms of integrating all necessary semantic data into the implementation and modelling the process dependencies. The latest research shows that synthetic event logs can also be generated by deep learning techniques. This has been demonstrated in a tool, called CoSMo (COnditioned Simulation MOdels) (Oyamada et al., 2023). CosMo uses Long short-term memory (LSTM) and Multilayer perceptron (MLP) to learn and model the constraints/conditions of process executions. The model then becomes the basis to generate synthetic data. Results show that CoSMo has achieved reasonable accuracy in recreating the input data from multiple business process datasets.

3. Background

3.1. What is an Event?

An event is a discrete record generated by the operating system or running application to record information of relevance for tasks such as future auditing, debugging, etc. Events often contain concise information relevant to the activity and are stored as either key-value pairs or as unstructured text data types. The amount of detail, format, and contents of the event differ depending on vendor/developer and specification. Figure 1 provides an example event description for assigning new file system permissions in Microsoft operating systems. As the text details, the event ID is 4670, and the series of objects that were involved or affected by the activity are listed below. This includes those related to the “Subject”, “Object”, “Process”, and those related to the “Permissions Change”. An example object is “Account Name” and the

object value is "John". On the other hand, Figure 2 presents an example event generated by the Linux system in the Syslog source. The event has been generated to report an invalid user admin login. Different from the Microsoft format, this event is without type and without a key-value definition. However, as the format is repeatable as it is based on a specific template, i.e., the account name always follows the 'user' keyword, it can be converted to a key-value representation.

3.2. Object-Centric Model

As event logs are recorded in many different formats, there is a need to have a common event-based data structure, that irrespective of the source, service, and vendor, can be stored and represented in a single, unified data structure for further processing. This requires implementing an individual pre-processing technique for each format (Marin-Castro & Tello-Leal, 2021); however, this is not the focus of this article.

In the majority of event logs, the format presents information in an object-centric manner. This is because operating systems themselves follow an object-centric philosophy whereby users, applications, data, etc. are all in a way data types and have values. Therefore, it is possible that each event can be converted to a common object-centric model, regardless of its format. As the focus of this paper is the generation of synthetic event datasets, we assume that the knowledge of how to parse both structured and unstructured logs (He et al., 2016) to extract meaningful objects, such as key-value pairs, is known. We acknowledge that there are open challenges in how an object model can be parsed without prior knowledge, handling missing data and heterogeneity; however, those challenges are not the focus of this article. There are several highly beneficial solutions available now that are capable of learning and extracting objects from the event log without human input (Zhu et al., 2019).

In this work, it is necessary to have a common event structure that is used to generate and store events. The common structure needs to be sufficiently flexible to store events extracted from different proprietary formats. Thereby we continue with the notation of events consisting of a series of objects, containing key-value pairs. In some event datasets, these key-value pairs are not explicitly described but are inherently there. For example, in event descriptions seen in Linux Syslog files, values relating to the keys of users, IP addresses, services, etc. are included. If we view the system in an object-oriented manner, then these values are objects. Pre-processing of such formats is required to translate them into a necessary key-value structure. However, this is only required for some definitions as the majority are already in a key-value structure. For example, as seen in Figure 1, in Microsoft events the key-value pairs are easily extracted, whereas in Figure 2, the objects are provided in a string that is more like a natural sentence.

The following object-centric representation is adopted in this paper, based on prior published works (Ghahfarokhi et al., 2021; Khan & Parkinson, 2018, 2019). The structure is as follows:

- D is used to model the set of event entries, where $D = \{E_1, E_2, ..., E_n\}$.
- The event, $E = \{T, ID, O\}$, where T is a timestamp, ID is a numeric event type ID, O is the set of objects, such that $O = \{O_1, O_2, ..., O_N\}$.
- Each entry, E_i, contains objects from O that are involved or effected by the event.

With real-time event logs, it should be noted here that where possible the 'ID' is taken directly from the event to establish type. However, with some event sources (e.g., Syslog), event-type information is not provided and one possible solution is to group events based on the information that is being presented. For example, the invalid user login attempt demonstrated in Figure 2 can be typed with all other event entries with the same description once the objects have been removed.

3.3. Event Chains

Event chains are a sequence of events grouped together as they are collectively part of the same user, application, or network activity or process. They denote a temporal sequence of events that are connected in terms of event information included within their contents. Figure 3 presents a real Microsoft event chain that involves 5 event types (4720, 4722, 4738, 4724, and 4726). This chain depicts one of the common approaches

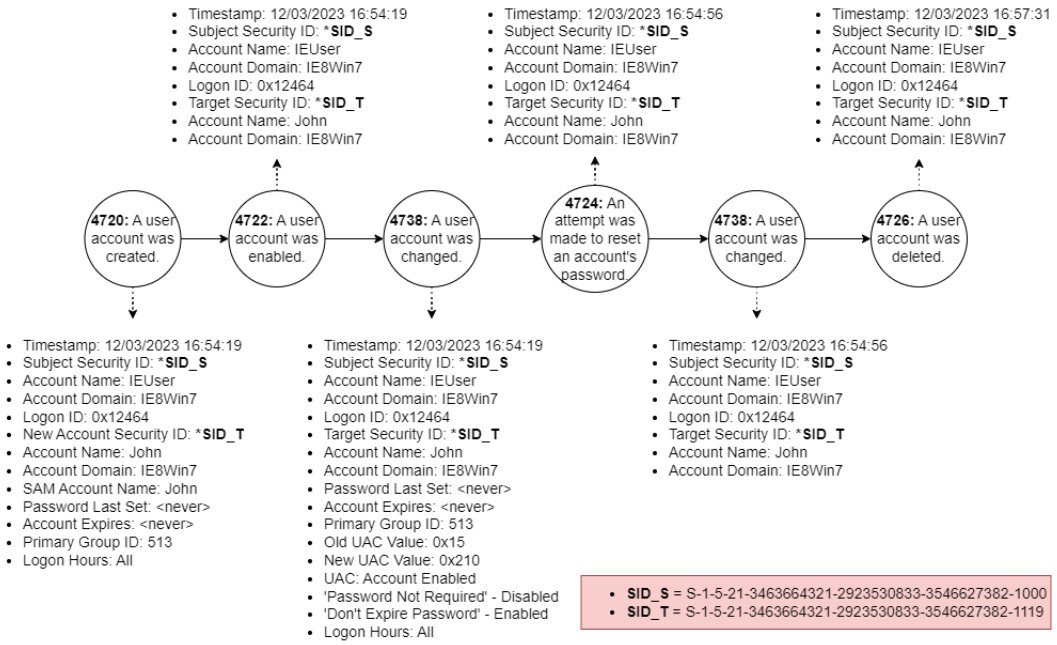


Figure 3: Example of a suspicious system activity. Note that both labels SID_S and SID_T are defined at the right-bottom of the figure

to concealing malicious activities, which is to create a user account, launch an attack through the new account and then delete it immediately to remove all data and traces (Al-Saleh & Al-Shamaileh, 2017).

It is immediately evident that the events in the chain share many common objects. In terms of object-centric representation, although the relationships between these event types may not be encoded, the events share descriptive content. More specifically, there is strong object co-occurrence among events. It is worth adding here that in the previous research related to mining event chains, researchers often refer to the number of shared objects as a measure of relationship strength (Khan & Parkinson, 2018). In the example, there are a total of 17 unique objects throughout the chain and all events share a common set of 7 objects, and there are only two events that have objects beyond the common 7 (4720 and 4738).

4. Dataset Generation

In this paper, we present a flexible tool for generating event log datasets. Note that, unlike the existing solutions, our proposed solution does not require any additional input. The only input required is the specification of 7 numeric parameters. The tool specifically generates an event log file by following an object-centric paradigm and contains noise and event chains (activities) discussed below. Figure 4 illustrates the process and is used to complement the following discussion.

4.1. Object Generation

In the first stage of the process, objects are generated for use and assignment to events. The number is dictated by the 'number of objects' parameter, and each object will be unique. The process does not create a key-value pair, which is because the motivation for this technique is to test and develop techniques capable of identifying event chains and we assume that the raw event log is constructed of key-value pairs or unstructured objects. It is assumed that in pre-processing the set of objects has been extracted, irrespective of structure. In the presented approach, the objects are named 'objectN' where 'N' is an incremental number assigned to each unique object.

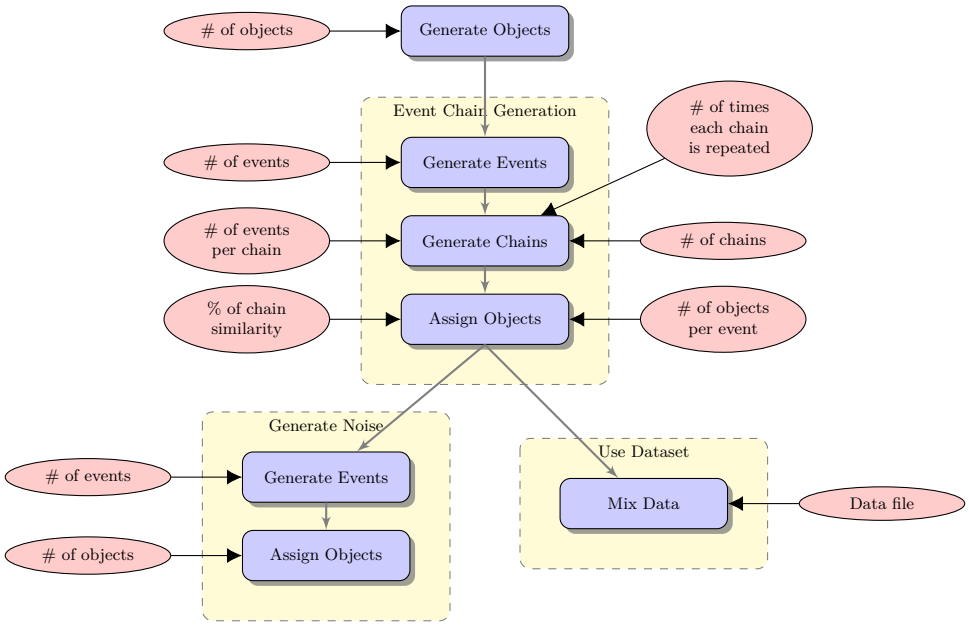


Figure 4: Synthetic Event Generation Process

4.2. Event Chain Generation

In the next stage, synthetic event chains are created. These chains are used to represent real event chains that are to be discovered in the dataset. Our generation process is based on the observation (explained in section 3.3) that the events in an event chain share a relatively high number of common objects in comparison to events that are not connected to each other. In other words, our tool generates an event chain by creating two or more events that have similar objects. Also note that the number of shared objects in any real-time event chain is not predictable and will change depending on the system generating the events, and contextual and situational information. Therefore, it is important to generate datasets with varying quantities of shared objects to capture the realistic event-logging mechanism.

In this work, the process of event chain generation has been broken down into three key stages. First, the events and event chains are generated depending on how many are required. Second, chains are generated depending on how many chains are required and how many events each requires (i.e., the chain length). In the third part, objects are assigned to the events in the chain. This is based on how many objects each event requires and the percentage similarity among events in the chain. This is used to control how many objects are the same across each event in the chain. For example, if each event has 10 objects and a similarity of 70% is specified, then 7 out of 10 objects in each event would be the same. As mentioned before, this mimics the behaviour/pattern of real-time event chains, where the events triggered by an activity have a number of common objects. Also, note that all event chains are labelled as incremental numeric values (0, 1, 2, ...), which enables their use as ground truth knowledge, essential for determining the correctness of clustering algorithms during the benchmarking process.

4.3. Generate Noise

In addition to the chains generated for the purpose of the section, the event dataset also needs to contain routine or benign events that represent normal system activity. In other words, events are not part of any event chain as they are often recorded for routine and generic descriptive information. In this work, similar to other works, these routine events are referred to as noise. The presented technique generates noise based on the number of events to generate and the number of objects to assign to each event. Unlike the event chains, events generated as noise do not have common objects or any pattern. Objects are taken from those

Parameter	Min	Max	Stepsize
Number Of event chains	10	40	10
Number of events in each chain	10	40	10
Number of objects in each event	5	15	5
Similarity % within chained events	20	80	20
Total number of events	10K	40K	10K
Total number of unique objects	100	400	100
Number of times each chain is repeated	1	4	1

Table 2: Parameter range for benchmark analysis.

created in the 'Generate Objects' activity earlier in the process before being randomly assigned to such events. The final event log dataset is created by combining the event chains and noise.

4.4. Using an Existing Data File

As an alternative to generating noise, the presented approach can also take an event file as input, before inserting the generated chains. This feature enables the end-user to take an event file and insert generated event chains, in order to provide an alternative mechanism of producing a complete event log dataset based on a real event dataset. There is a prerequisite here that the input file has been converted into the required event-object model, as presented in Section 3.2. The purpose of functionality is to enable users to insert event chains for discovery in real-world datasets, which may have different characteristics (in particular in terms of noise) than those systematically generated. The distribution of noise is often dependent on the system generating the event dataset, and researchers and developers researching a solution for a specific application will generate more realistic results by using real datasets.

4.5. Specification

In this work, we generate a range of event datasets following a pragmatic approach to demonstrate how the generator can work and to provide some initial benchmarks for further analysis. Table 2 provides the specific range for each of the input parameters used to generate this benchmark analysis. These ranges were selected to align with prior work identified in Section 2 and also to accommodate available resource requirements. However, we would like to emphasise that the event generator is flexible and can generate event datasets much larger than the ones used in this research. In this work, we generate and benchmark more than 12,000 files.

5. Clustering algorithms for Benchmarking

Following algorithms are selected to evaluate their ability in identifying event chains in the datasets generated by the proposed event data generator: DBSCAN (Density-Based Spatial Clustering of Applications and Noise) (Ester et al., 1996), OPTICS (Ordering Points To Identify the Clustering Structure) (Ankerst et al., 1999), Affinity Propagation (Abdulah et al., 2022), Mean Shift (Zhu et al., 2022) and Agglomerative (Han et al., 2023). Our aim is to determine how accurate and time-efficient these algorithms are in separating and finding the systematically inserted event chains from noise, for benchmarking purposes. The selected clustering algorithms are those that do not require any user input, such as pre-specifying the number of clusters or size of the neighbourhood. Moreover, we performed several experiments to tune hyperparameter values in order to reduce the loss and obtain the best possible performance these algorithms could achieve. The following sections provide information on the algorithm and its configurations, determined by preliminary experimentation. These configurations are necessary to replicate our results.

6. Experimentation and Evaluation Metric

The event data generator is developed in C# programming language. The benchmarking experiments are performed using the clustering algorithms implemented in the well-known scikit-learn Python library on a 28-core CPU Intel(R) Xeon(R) Platinum 8180 @ 2.50GHz with 128GB RAM.

To evaluate the performance of the clustering methods, we have employed the ARI (Adjusted Rand Index), which is an adjusted version of Rand Index (Hubert & Arabie, 1985). First, we present how to calculate Rand Index, which is as follows:

$$RI = \frac{TP+TN}{TP+TN+FP+FN}$$

The TP (True positive) is the number of pairs of event log entries that are in the same subset in the predicted event chain and in the same subset in the ground-truth (i.e., actual) event chain, which is provided in the labelled input dataset as mentioned before. TN (True negative) is the number of pairs of event entries that are not in the same subset in the predicted event chain and not in the same subset in the ground-truth event chain. FP (False positive) is the number of pairs of data points that are in the same subset in the predicted event chain but not in the same subset in the ground-truth event chain. FN (False negative) is the number of pairs of data points that are not in the same subset in the predicted event chain but in the same subset in the ground-truth event chain. The value of RI ranges from 0 to 1, where 1 means the predicted and actual event chains match identically.

RI suffers from a major drawback in that it is sensitive to chance (D'Ambrosio et al., 2021). This means, if the input data is large enough, it is possible that some data points will be randomly grouped together in a correct manner, so the RI might never actually be 0. The Adjusted Rand Index (ARI) considers this possibility (Abdullayeva, 2022), so it adjusts/rescales the Rand Index and presents the percentage of correct predictions among all predictions (Sinnott et al., 2016). ARI is calculated as follows:

$$ARI = \frac{RI - Expected_RI}{max(RI) - Expected_RI}$$

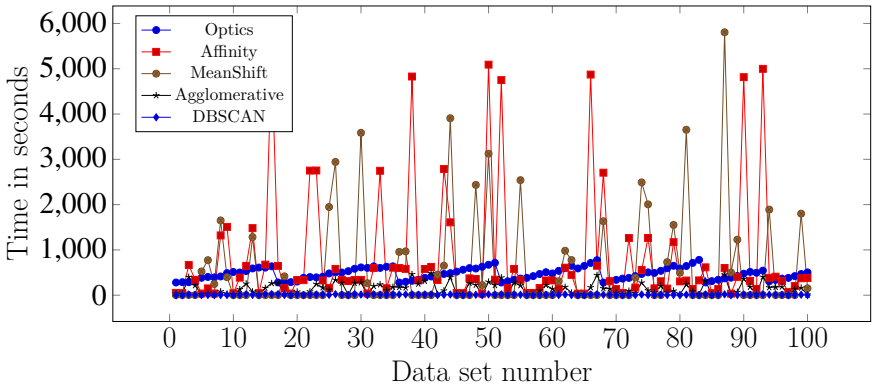
Where Expected_RI is the expected value of RI when the clusters (i.e., event chains in our case) are made by coincidence but while keeping the same marginal distributions and max(RI) is always 1. ARI value ranges from -1 to 1, where 1 indicates a perfect match between the two clusters, 0 indicates a random match, and -1 indicates that the two clusters do not match at all.

7. Preliminary Experiments

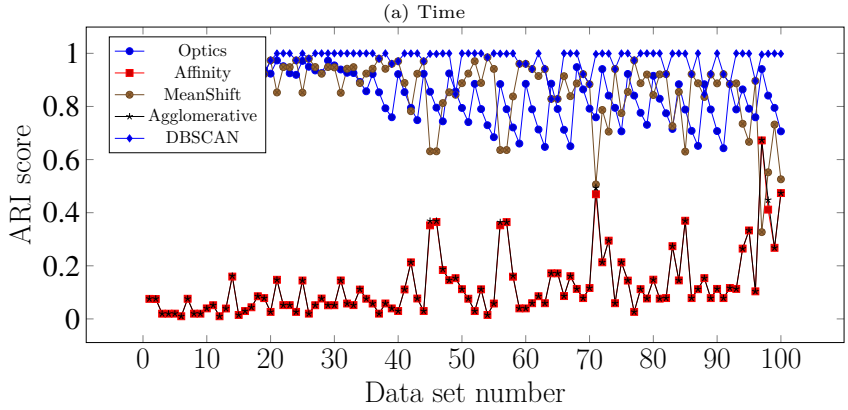
Preliminary experiments took place to establish the capabilities of the algorithms and to determine if we are able to realistically acquire a full series of results for all algorithms within a reasonable time frame using available computing resources. Soon after starting the experiment, we realised that it would not be possible to run all algorithms for every dataset as it would take a significant amount of time. Therefore, we decided to use a small subset of the dataset in order to determine the performance of the algorithms and determine which should be considered for use to generate a full series of results for all datasets. This preliminary experiment is performed by selecting 100 event log samples that are representative of the generated dataset and processing them with each of the five algorithms mentioned in section 5.

7.1. Capability

Figure 5b presents the Adjusted Rand Index (ARI) of all five algorithms on 100 data files and clearly shows DBScan as the best algorithm based on the ARI Score. The agglomerative clustering algorithm has shown the worst performance as the highest ARI obtained is around 0.7 and the lowest is almost 0. The affinity algorithm also obtained results as poor as Agglomerative. For the MeanShift algorithm, only a few event log files obtained an ARI score as good as DBSCAN. Regarding Optics, the results are mostly poorer than the MeanShift as well as the DBSCAN algorithm. In summary, DBSCAN is identified as consistently best.



(a) Time



(b) Adjusted Rand Index

Figure 5: Performance results from 5 selected clustering algorithms using a small dataset

7.2. Execution time

Figure 5a illustrates the comparison of execution time in seconds by running the selected algorithms on the 100 samples. It is evident that DBSCAN is the fastest algorithm. Agglomerative is second best but gives the worst ARI score as discussed above. The next best algorithm is OPTICS; however, as it can be seen in Figure 5b, the corresponding ARI values obtained are poor when compared to the DBSCAN. MeanShift consumes a lot of time for each file when compared to DBSCAN and the corresponding ARI scores are also not satisfactory. The Affinity algorithm has taken the maximum amount of time available before the algorithm times out, and the ARI values are the lowest.

Therefore, based on the results obtained as discussed above, DBSCAN has been selected for performing benchmark analysis on the generated datasets.

8. Results and Discussion

The following section presents the benchmarking results and their detailed analysis from running DBSCAN on all generated datasets. The section is divided into 3 parts. First, the importance of each feature is determined to understand the relationship between the features and the Adjusted Rand Index (ARI) score. This shows which features are relevant and can greatly influence the classification accuracy of DBSCAN. Second, we present and discuss the range of ARI scores for each individual feature, when considering different input values. This provides further insight into the algorithm's strengths and weaknesses. Finally, we present how multiple features with various input values impact the overall classification accuracy of the algorithm.

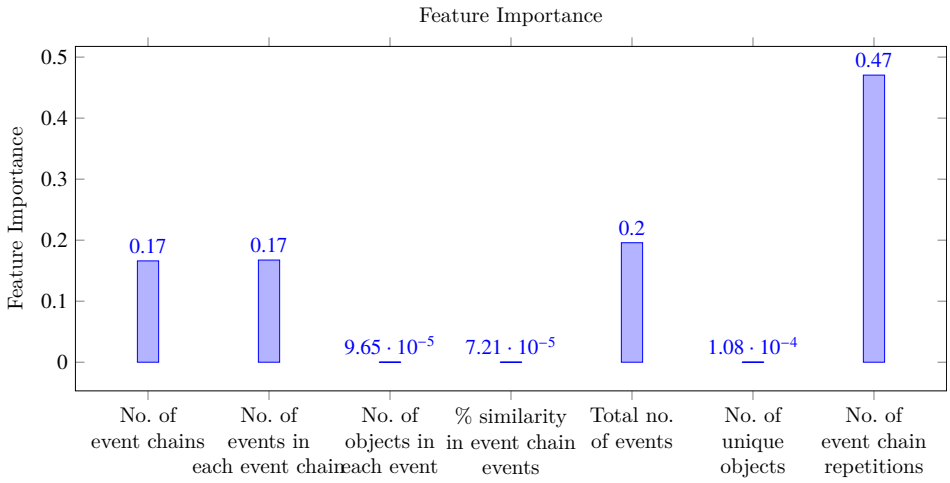


Figure 6: Feature Importance score of all synthetic event generation parameters.

8.1. Determining Feature Importance

Based on existing research, we have utilised the Random Forest (RF) Regressor algorithm to evaluate the importance of each feature present in our generated dataset. Although many feature selection techniques are available, such as those based on linear regression, logistic regression, and neural networks, RF Regressor has been chosen because of its speed of computation, better accuracy, and control over-fitting (Yu et al., 2020; Alkuhlani et al., 2022; Saputra et al., 2019). As the RF Regressor is based on ensemble learning, it finds the 'contribution value' of every feature in every (permuted) decision tree running in the model using the Mean Squared Error (MSE) loss function and then averages all values together leading to a stronger prediction. A feature is deemed unimportant if the MSE remains high and unchanged over a given number of iterations (Grömping, 2015).

Figure 6 presents the feature importance of each attribute corresponding to the target variable (ARI score). The sum of all the feature importance numeric values computed by the RF Regressor is 1, the feature with the highest numeric value is considered most important and the second highest is considered second most important, and so on. It can be seen from the graph that the following 3 features of our dataset have no significant impact on the ARI score: the total number of unique objects, percentage similarity in event chain events, and the number of objects in each event. The remaining 4 features are considered significant by the RF Regressor algorithm. The number of times each event chain is repeated feature has the highest importance value of 0.47. The number of event chains and the number of events in each event chain have a similar influence on the ARI score. The total number of events feature is slightly more effective in terms of the numeric value of nearly 0.2 on ARI.

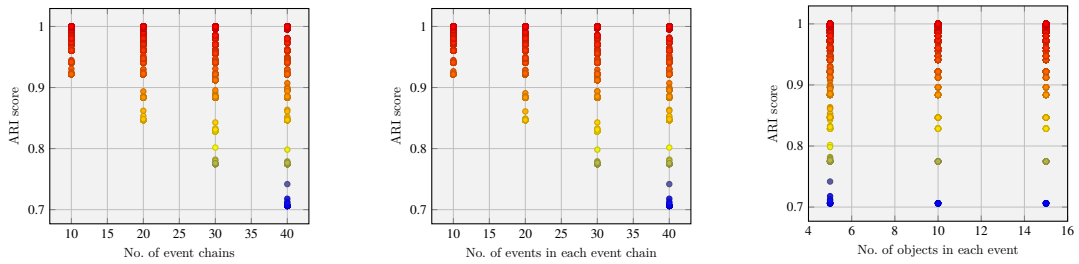
8.2. Analysis of Each Individual Feature

The acquired ARI score against each individual feature is shown in Figure 7 and discussed in the following text. As mentioned in Section 6, the ARI has been employed as the performance evaluation metric in this work. The value of ARI ranges from 0 to 1, the higher the value the more accurate the grouping of data points in the datasets. The value of $ARI \geq 0.95$ is considered optimal (Wang et al., 2012). In addition to the following discussion, the reader is directed to Table 3 for the full benchmark values.

Both the number of event chains and the number of events in each event chain have a similar impact on the ARI score as evident from the graphs shown in Figures 7a and 7b, respectively. Both features have the same numerical values (10, 20, 30, and 40) accordingly, and have the same degree of importance. For both features, value 40 has scored a relatively low ARI, with a minimum of 0.71. The value of 10 has achieved the maximum average ARI, with a minimum of 0.91. For values 20 and 30, the minimum scored ARI is 0.85 and 0.78, respectively. These particular results indicate that increasing the values of these features has a

Variable	Min	Lower Quartile	Median	Upper Quartile	Max
Number of Event Chains					
10	0.921752	0.999998	0.997334	0.999854	0.999963
20	0.846705	0.999996	0.995121	0.999722	0.999928
30	0.774857	0.999993	0.993490	0.999604	0.999893
40	0.706211	0.999991	0.980102	0.999479	0.999859
Number of events in each chain					
10	0.921632	0.999998	0.995007	0.999941	0.999983
20	0.846545	0.999991	0.990277	0.999768	0.999928
30	0.774737	0.999979	0.985062	0.999506	0.999841
40	0.706211	0.999962	0.980110	0.999161	0.999722
Number of objects in each event					
5	0.706211	0.999998	0.995007	0.999681	0.999921
10	0.706211	0.999998	0.995007	0.999681	0.999921
15	0.706211	0.999998	0.995007	0.999681	0.999922
Percentage similarity in each event chain					
20	0.706211	0.999998	0.995007	0.999681	0.999922
40	0.706211	0.999998	0.995007	0.999681	0.999921
60	0.706211	0.999998	0.995007	0.999681	0.999921
80	0.706211	0.999998	0.995007	0.999681	0.999922
Total number of events					
1000	0.706211	0.999965	0.991708	0.998724	0.999555
2000	0.846552	0.999991	0.996514	0.999636	0.999877
3000	0.896245	0.999996	0.997785	0.999830	0.999944
4000	0.921638	0.999998	0.995007	0.999908	0.999972
Number of unique objects					
100	0.706211	0.999998	0.995007	0.999684	0.999921
200	0.706211	0.999998	0.994176	0.999681	0.999922
300	0.706211	0.999998	0.995007	0.999681	0.999921
400	0.706211	0.999998	0.995007	0.999681	0.999921
Number of event chain repetitions					
1	0.706211	0.995307	0.921638	0.960409	0.980103
2	0.998145	0.999998	0.999759	0.999922	0.999969
3	0.996777	0.999995	0.999490	0.999838	0.999940
4	0.995442	0.999991	0.999176	0.999707	0.999875

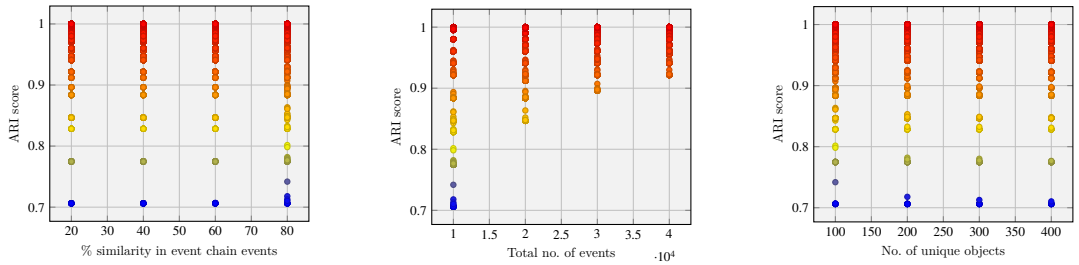
Table 3: Benchmark results showing the ARI measurement in relation to the changing input variables



(a) The number of event chains and ARI score

(b) The number of events in each event chain and ARI score

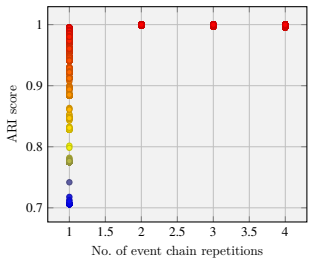
(c) The number of objects in each event and ARI score



(d) Percentage similarity in event chain events and ARI score

(e) The total number of events and ARI score

(f) The number of unique objects and ARI score



(g) Number of times each event chain is repeated and ARI score

Figure 7: Comparison of ARI scores from every individual parameter used for synthetic event generation.

negative impact on the clustering capability of the DBSCAN algorithm. In other words, small event chains with a small number of events result in reduced classification accuracy.

The following 3 features have an almost similar impact on the ARI score. These are the number of objects in each event, percentage similarity in event chain events, and number of total unique objects. This is evident in the graphs shown in Figures 7c, 7b and 7f for each of the features, respectively. As previously identified, these features were found to be the least relevant in our generated dataset. The ARI score obtained by these 3 features ranges from 0.71 to 0.99 and it cannot be clearly specified which feature value(s) always result in poor ARI scores. However, it should be noted that these results, although being identified to be of least significance, still demonstrate that the greater these values are, the better the ARI score. This clearly indicates that increasing the number of objects in each event, increasing the similarity within event chains, and the number of total events all result in the algorithm being more accurate at correctly classifying the event chains.

Figure 7e presents the relationship between the different total numbers of unique events and their corresponding ARI scores. The maximum average ARI score is achieved in data files that contain 40,000 events, with a low of 0.91 and a high of 0.99 ARI. For data files containing 20,000 and 30,000 events, their minimum ARI scores are 0.85 and 0.90, respectively. The lowest average ARI score is obtained in those data files where there are only 10,000 events. As previously identified, this is the second most important feature, so increasing the number of events has a positive impact on the ARI scores. This demonstrates that an

increasing number of unique objects in the event dataset has a positive impact on the ability to correctly identify event chains. This is somewhat to be expected, as more unique objects will result in more easily differentiated events.

The ARI produced by changing the number of times each event chain is repeated is shown in Figure 7g. It is clear that when the value is 1, the average ARI score is the lowest, with a minimum of 0.70. However, when the value is 2 or above, the average ARI score becomes the highest, which is around 0.99. This feature has the highest importance in our dataset, so it could be determined that if an event chain is repeated multiple times, it becomes easier for the DBSCAN algorithm to accurately identify it.

Based on the above results and discussion, it can be concluded that certain individual features influence the accuracy of DBSCAN. However, this might not be a complete picture as the clustering process is performed using all 7 features combined. Therefore, it is important to determine how these features impact the ARI score collectively, which is presented in the next section.

8.3. Consolidated Analysis of Features

In this section, the 4 most important features in our generated dataset (number of event chains, number of events in each event chain, total number of events and number of times each event chain is repeated) are analysed together to determine which of their values give the poor ARI score. This will enable us to gain further insight into the benchmarking results and uncover any general pattern (in terms of dataset specification) that might lead to poor results.

The combinations of all features values, where the ARI scores are less than optimal (i.e., below 0.95), are determined manually by the following process:

- List all feature values based on the specification shown in Table 2 and their ARI score obtained by performing DBSCAN on the corresponding data files.
- Remove all rows from the analysis where the ARI score is 0.95 or above.
- Sort the list on ARI score, from lowest to highest.
- Group rows together where the 4 important features have the same values. As each group denotes multiple data files due to having different values of the irrelevant features, it contains a range of different ARI scores.
- All such groups and the corresponding ARI score range are shown in Table 4, along with the average ARI. For example, in a case where the number of event chains is 40, the number of events in each event chain is 40, the total number of events is 10,000 and the number of times each event chain is repeated is 1, the ARI score ranges from 0.706 to 0.737, averaging at 0.726.

Note that the feature values not shown in Table 4 have achieved 0.95 or above ARI, which is considered an optimal score, and therefore excluded from the analysis. The reason we are examining the poor results only is that they are fewer in number, which makes it easier to separate and analyse.

Based on the table, it is evident that the worst ARI score is obtained in those event log files where the number of event chains and number of events in each event chain are the highest (i.e., 40) and the total number of events are the lowest (i.e., 10,000). So if a number of large event chains that are usually representative of a number of lengthy users, system, network, or application activities are densely packed in a relatively smaller event log dataset, it reduces the accuracy when clustering. This is most likely due to not having clear distinction or separation among the event chains (clusters) boundaries (Xu & Tian, 2015). If there are multiple event chains that are similar in terms of event objects and overlap without a drop in density, they are forced into a single event chain, thereby producing poor results. Another observation is regarding the number of times each event chain is repeated, which is always 1 in case of all poor results. So if a certain activity is recorded only once within a particular event dataset, it becomes difficult for the clustering algorithm to detect. Finally, there are 1,536 files out of the 12,288 generated that have obtained less than the optimal ARI score (0.95). In other words, only 12.5% of the generated dataset has obtained an ARI score that is less than 0.95. An interesting observation from the results is that the number of files for

Num of event chains	Num of events in each event chain	Total number of events	Num of event chain repetitions	Average ARI	Range of ARI scores
40	40	10000	1	0.726	0.7026-0.736
40	30	10000	1	0.7746	0.774–0.776
30	40	10000	1	0.7809	0.7748-0.8010
30	30	10000	1	0.8285	0.828-0.8428
40	20	10000	1	0.8465	0.8465-0.8467
40	40	20000	1	0.8477	0.8465-0.8635
20	40	10000	1	0.85	0.8467-0.8619
40	40	20000	1	0.8563	0.8465-0.8635
40	30	20000	1	0.8836	0.8836-0.893
30	20	10000	1	0.89	0.8837-0.8986
30	40	20000	1	0.891	0.8837-0.8955
20	30	10000	1	0.887	0.8837-0.8900
40	40	30000	1	0.90	0.8962-0.9071
30	30	20000	1	0.91	0.9120-0.9186
30	30	10000	1	0.91	0.9120-0.9153
40	10	10000	1	0.921	0.9216-0.9216
40	20	20000	1	0.9216	0.92163-0.9217
40	30	30000	1	0.92637	0.9216-0.9299
40	40	40000	1	0.921	0.9216-0.92165
30	40	30000	1	0.92651	0.9216-0.9288
20	20	10000	1	0.9267	0.92167-0.9299
20	40	20000	1	0.9299	0.9217-0.9306
10	40	10000	1	0.94	0.940-0.940
40	30	40000	1	0.94	0.9409-0.944
30	10	10000	1	0.94	0.940-0.9453
30	20	20000	1	0.9409	0.945-0.945
30	30	30000	1	0.9409	0.940-0.944
30	40	40000	1	0.940929	0.940-0.940
20	30	20000	1	0.940943	0.940-0.9409
10	30	10000	1	0.9440	0.940–.9449
40	20	30000	1	0.948	0.947-0.950
20	40	30000	1	0.949	0.94-0.9499

Table 4: The combination of the 4 most significant features that resulted in poor ARI scores (< 0.95)

each combination/group of feature values resulting in less than optimal score is always 48. Also, note that no log file in the dataset obtained less than 0.7 ARI, which is also considered to be a good score.

Considering the benchmark results and the discussion above, it has been established that the application of clustering is suitable for finding and extracting event chains from a noisy event log data set. Depending on the algorithm, the clustering approach produces accurate results in most cases and is time efficient. DBSCAN has demonstrated that it is able to efficiently produce good results.

9. Conclusion

This research focuses on developing and describing a tool capable of generating event datasets for the development and analysis of security techniques processing event datasets. The motivation for this work is because in previous work, researchers either use a synthetically generated dataset or a real-world dataset with the availability of ground-truth knowledge. These datasets are appropriate for their research, but due to security sensitivity, they are often not shared openly, making it challenging to draw direct comparisons between techniques. In this work, we present a parameterised generator application capable of generating datasets for a wide range of research objectives centred on the use of event chains.

The technique we have produced is defined by 7 parameters to define characteristics of the generated event log, including information about its size, objects, and relationships. Further, parameters are used to define the characteristics of the event chain. In addition to generating synthetic datasets, the technique can also use real event datasets as a base for adding synthetic chains with ground-truth knowledge. This combination ensures the generator is flexible and useful for wide-ranging application scenarios.

We then perform benchmarking on a range of datasets, generated by considering different parameter combinations within a predefined range. The work in this paper focuses on the use of clustering to detect event chains. The initial analysis demonstrated that DBSCAN is the most appropriate algorithm to use in benchmarking as it can handle the generated datasets in terms of size and processing time. DBSCAN is then applied to all generated datasets, generating promising results and highlighting that clustering provides a suitable approach to event chain detection. This paper and the event generation application provide the foundation for future research in this discipline, with a flexible generation technique and datasets that can be shared.

10. Availability

The event generator and data files used in this research are readily available from the authors. Due to their size and funding mandate, we will release them on request. Please email to request a copy.

11. Acknowledgement

The authors would like to express their gratitude for the financial support from The UK's Defence Science and Technology Laboratory (Dstl) provided through the UK's Defence and Security Accelerator (DASA) and Managed by Frazer-Nash Consultancy.

References

Abdulah, S., Atwa, W., & Abdelmoniem, A. M. (2022). Active clustering data streams with affinity propagation. ICT Express, 8, 276–282.

Abdullayeva, F. J. (2022). Distributed denial of service attack detection in e-government cloud via data clustering. Array, 15, 100229.

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.

Alkuhlani, A., Gad, W., Roushdy, M., & Salem, A.-B. M. (2022). O-glycosylation site prediction using randome forest importance and support vector machine. In 2022 IEEE 3rd International Conference on System Analysis & Intelligent Computing (SAIC) (pp. 1–5). IEEE.

Alshaikh, O., Parkinson, S., & Khan, S. (2022). On the variability in the application and measurement of supervised machine learning in cyber security. In Intfernational Conference on Ubiquitous Security (pp. 545–555). Springer.

Alzhrani, F., Saeedi, K., & Zhao, L. (2023). Bela: A blockchain event log app, .

Ankerst, M., Breunig, M., Kriegel, H., & Sander, J. (1999). Acm sigmod record. New York: ACM, (pp. 49–60).

Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Maggi, F. M., Marrella, A., Mecella, M., & Soo, A. (2018). Automated discovery of process models from event logs: review and benchmark. IEEE transactions on knowledge and data engineering, 31, 686–705.

Buijs, J. (2014). Receipt phase of an environmental permit application process ('wabo'), coselog project. Eindhoven University of Technology, .

Burattin, A. (2015). Plg2: multiperspective processes randomization and simulation for online and offline settings. arXiv preprint arXiv:1506.08415, .

Burattin, A., Re, B., Rossi, L., & Tiezzi, F. (2022). A purpose-guided log generation framework. In Business Process Management: 20th International Conference, BPM 2022, Münster, Germany, September 11–16, 2022, Proceedings (pp. 181–198). Springer.
Chen, Z., Liu, J., Gu, W., Su, Y., & Lyu, M. R. (2021). Experience report: Deep learning-based system log analysis for anomaly detection. arXiv preprint arXiv:2107.05908, .
Cinque, M., Della Corte, R., & Pecchia, A. (2020). Contextual filtering and prioritization of computer application logs for security situational awareness. Future Generation Computer Systems, 111, 668–680.
De Leoni, M., & Mannhardt, F. (2015). Road traffic fine management process. Eindhoven University of Technology, Dataset, 284.
Delias, P., Doumpos, M., Grigoroudis, E., Manolitzas, P., & Matsatsinis, N. (2015). Supporting healthcare management decisions via robust clustering of event logs. Knowledge-Based Systems, 84, 203–213.
D'Ambrosio, A., Amodio, S., Iorio, C., Pandolfo, G., & Siciliano, R. (2021). Adjusted concordance index: an extensionl of the adjusted rand index to fuzzy partitions. Journal of Classification, 38, 112–128.
Esgin, E., & Karagoz, P. (2019). Process profiling based synthetic event log generation. In KDIR (pp. 516–524).
Ester, M., Kriegel, H.-P., Sander, J., Xu, X. et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In kdd (pp. 226–231). volume 96.
Fahrenkrog-Petersen, S. A., van der Aa, H., & Weidlich, M. (2023). Optimal event log sanitization for privacy-preserving process mining. Data & Knowledge Engineering, 145, 102175.
Ghahfarokhi, A. F., Park, G., Berti, A., & van der Aalst, W. M. (2021). Ocel: A standard for object-centric event logs. In European Conference on Advances in Databases and Information Systems (pp. 169–175). Springer.
Grömping, U. (2015). Variable importance in regression models. Wiley interdisciplinary reviews: Computational statistics, 7, 137–152.
Grüger, J., Geyer, T., Jilg, D., & Bergmann, R. (2023). Sample: A semantic approach for multi-perspective event log generation. In Process Mining Workshops: ICPM 2022 International Workshops, Bozen-Bolzano, Italy, October 23–28, 2022, Revised Selected Papers (pp. 328–340). Springer.
Han, X., Zhu, Y., Ting, K. M., & Li, G. (2023). The impact of isolation kernel on agglomerative hierarchical clustering algorithms. Pattern Recognition, 139, 109517.
He, P., Zhu, J., He, S., Li, J., & Lyu, M. R. (2016). An evaluation study on log parsing and its use in log mining. In 2016 46th annual IEEE/IFIP international conference on dependable systems and networks (DSN) (pp. 654–661). IEEE.
He, S., He, P., Chen, Z., Yang, T., Su, Y., & Lyu, M. R. (2021). A survey on automated log analysis for reliability engineering. ACM computing surveys (CSUR), 54, 1–37.
Huang, S., Liu, Y., Fung, C., He, R., Zhao, Y., Yang, H., & Luan, Z. (2020). Paddy: An event log parsing approach using dynamic dictionary. In NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium (pp. 1–8). IEEE.
Hubert, L., & Arabie, P. (1985). Comparing partitions. Journal of classification, 2, 193–218.
Kara, I. (2021). Read the digital fingerprints: log analysis for digital forensics and security. Computer Fraud & Security, 2021, 11–16.
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. URL: https://www.sciencedirect.com/science/article/pii/S2214212619303060. doi:https://doi.org/10.1016/j.jisa.2019.102375.
Khan, S., & Parkinson, S. (2022). Identifying high-risk over-entitlement in access control policies using fuzzy logic. Cybersecurity, 5, 1–17.
Li, T. (2015). Event mining: algorithms and applications volume 38. CRC Press.
Mannhardt, F. (2017). Hospital billing-event log. Eindhoven University of Technology. Dataset, (pp. 326–347).
Mannhardt, F. et al. (2016). Sepsis cases-event log. Eindhoven university of technology, 10.
Marin-Castro, H. M., & Tello-Leal, E. (2021). Event log preprocessing for process mining: a review. Applied Sciences, 11, 10556.
Oliner, A., & Stearley, J. (2007). What supercomputers say: A study of five system logs. In 37th annual IEEE/IFIP international conference on dependable systems and networks (DSN'07) (pp. 575–584). IEEE.
Oyamada, R. S., Tavares, G. M., & Ceravolo, P. (2023). Cosmo: a framework for implementing conditioned process simulation models. arXiv preprint arXiv:2303.17879, .
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.
Parkinson, S., Vallati, M., Crampton, A., & Sohrabi, S. (2018). Graphbad: A general technique for anomaly detection in security information and event management. Concurrency and Computation: Practice and Experience, 30, e4433.
Rai, P. et al. (2011). Data clustering: K-means and hierarchical clustering. CS5350, 6350.
Saputra, E. S., Putrada, A. G., & Abdurohman, M. (2019). Selection of vape sensing features in iot-based gas monitoring with feature importance techniques. In 2019 Fourth International Conference on Informatics and Computing (ICIC) (pp. 1–5). IEEE.
Sinnott, R., Duan, H., & Sun, Y. (2016). Chapter 15—a case study in big data analytics: exploring twitter sentiment analysis and the weather. Big Data, (pp. 357–388).
Špaček, S., Velan, P., Čeleda, P., & Tovarňák, D. (2022). Encrypted web traffic dataset: Event logs and packet traces. Data in

Brief, 42, 108188.
Steeman, W. (2013). Bpi challenge 2013. Ghent University, Dataset, .
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).
Suriadi, S., Andrews, R., ter Hofstede, A. H., & Wynn, M. T. (2017). Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. Information systems, 64, 132–150.
Turcotte, M. J., Kent, A. D., & Hash, C. (2019). Unified host and network data set. In Data Science for Cyber-Security (pp. 1–22). World Scientific.
Vaarandi, R. (2003). A data clustering algorithm for mining patterns from event logs. In Proceedings of the 3rd IEEE Workshop on IP Operations & Management (IPOM 2003)(IEEE Cat. No. 03EX764) (pp. 119–126). Ieee.
Wang, H.-M., Hsiao, C.-L., Hsieh, A.-R., Lin, Y.-C., & Fann, C. S. (2012). Constructing endophenotypes of complex diseases using non-negative matrix factorization and adjusted rand index. PLoS One, 7, e40996.
Wei, J., He, Z., Ouyang, C., & Moreira, C. (2022). Mimicel: Mimic-iv event log for emergency department, .
Wu, S., Wang, B., Wang, Z., Fan, S., Yang, J., & Li, J. (2022). Joint prediction on security event and time interval through deep learning. Computers & Security, 117, 102696.
Wurzenberger, M., Skopik, F., Settanni, G., & Scherrer, W. (2016). Complex log file synthesis for rapid sandbox-benchmarking of security-and computer network analysis tools. Information Systems, 60, 13–33.
Xu, D., & Tian, Y. (2015). A comprehensive survey of clustering algorithms. Annals of Data Science, 2, 165–193.
Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. IEEE Transactions on neural networks, 16, 645–678.
Yu, J., Xia, C., Xie, J., & Zhang, H. (2020). Research on feature importance of gait mechanomyography signal based on random forest. In 2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL) (pp. 191–196). IEEE.
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.
Zhu, T., Wang, X., Zhang, J., Yu, S., & Molotov, I. (2022). Mean-shift clustering approach to the tracklets association with angular measurements of resident space objects. Astronomy and Computing, 40, 100588.
Zisgen, Y., Janssen, D., & Koschmider, A. (2022). Generating synthetic sensor event logs for process mining. In Intelligent Information Systems: CAiSE Forum 2022, Leuven, Belgium, June 6–10, 2022, Proceedings (pp. 130–137). Springer.