Automated learning and exploitation of action-based domain models from security event data

Delivery Report

Simon Parkinson, Saad Khan, Monika Roopak. Department of Computer Science, University of Huddersfield. s.parkinson@hud.ac.uk

Executive Summary

Event-based data sources are an essential part of cyber defence monitoring. They describe the status of the system, and their interpretation is essential to gaining an understanding of system security so that threats can be acted upon immediately; however, identifying such information is challenging. Manual approaches are restricted by operator capacity and automated techniques are reliant on the presence and inclusion of knowledge. This project focuses on developing techniques to identify action-based knowledge models from underlying security data, to perceive the environment through algorithms capable of self-learning structure in event-based data sources, before generating and enabling the exploitation of action-based representations. In this project, an automated approach to identify events related to a system security activity (herein called event chains) in event-based datasets has been developed. These event chains, with the assistance of human input, can then be translated into an action model. The action model can then be used to automate the identification and response to future event-chain occurrences. The data generation aspect of the project proved to be more important than anticipated, largely due to the absence of any publicly available dataset approach to generate event-based datasets with diverse characteristics.

Contents

Project Background

Deliberation is required in most data-rich environments, to understand the perceived environment and to generate strategies (sequence of actions) to achieve the desired outcome. Such deliberation typically involves a combination of domain experts and knowledge stored and used within software tools to first process available data sources to extract a meaningful data subset (either manually or through computational tools), often followed by the generation of one or more potential courses of actions. Consider cyber analysis as an example, vast quantities of real-time data are available which provide an instantaneous 'snapshot' of the activity within the monitored system.

Such data can be used by software agents to respond to known situations based on the predefined knowledge base. Software agents can deliberate using knowledge and the system snapshot. Relevant to this proposal, this could be based on an action-based model whereby specific actions can be executed based on the system snapshot and desired response. Software agents are very effective at responding to known incidents, where knowledge and mitigation strategies have been encoded. However, this has created a new bottleneck – that of knowledge generation – where experts capture, encode and distribute knowledgebases to be used by autonomous agents. This project focuses on the development and use of techniques capable of learning domain, action-based knowledge from monitoring system activity for automatic generation, allowing its exploitation of response strategies.

In this project, techniques have been developed and tested that can identify event chains, before aiding the creation of action-based models from event log sources, without using any predefined knowledge. The approach works by extracting an object-based representation and identifying strong connections between events. The object-based representation allows the detection of changes in the system through their relationship with different event types (security warnings, configuration changes, etc.). The learned action model is structured in terms of precondition – a prior system state before a change happened – and an effect that describes the system change. The learned action model is then used alongside automated deliberation techniques to automatically predict and suggest actions to the operator where the same precondition is identified and/or the desired effect is required.

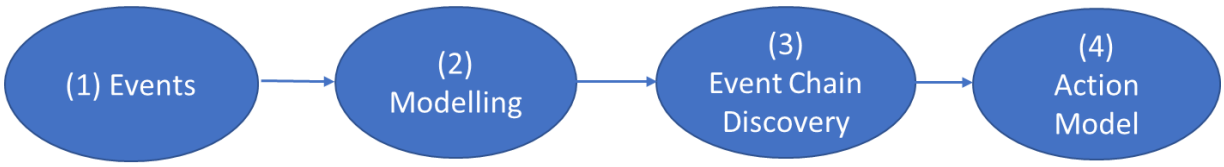


FIGURE 1 - PROJECT DATA/FUNCTION FLOW

Figure 1 illustrates the separate functional areas of the overall project and their ordering and dependency. First, event-based datasets and their single input to the system. There is no requirement over what system generated them, only that they have been saved in comma-separated value format. This ensures the technique is vendor agnostic; however, some event data sources that are not in a common key-value (e.g., Microsoft) or value (e.g., syslog) format will require preprocessing. After parsing, the event-based datasets are to be converted into an internal model appropriate to store the event data, allowing it to be exploited to identify event chains, which are a sequence of closely linked events that are relevant to the same security activity within the system. Finally, the event chain can be converted into an action for future automation purposes. An action would have a precondition and effect. The precondition represents the events that occur before something happens and requires operator assistance to separate precondition and effect events.

Summary of Main Deliverables

All code deliverables and publications are available at the following GitHub repository:
https://github.com/sparkins01/ARCD

Software:

1. **Data Generator** capable of generating synthetic event datasets with event chains for future discovery. The event chain specification is provided through command line parameters. An explanation of the work undertaken and included in this command line application is in the section entitled WP1: Establishment and Generation of Key Data Sources.
2. **Event-chain detection clustering algorithms** (Python scripts) were used in the experimental stage of this project to determine the most suitable clustering algorithm for event-chain detection. Empirical results demonstrated DBSCAN as the most suitable. An explanation of the work undertaken and included in the Python script is in the section entitled WP2: Knowledge Discovery, Learning, and Formulation.
3. **Prototype application** showing how (1) and (2) can be utilised to identify event chains that are then configured into an action for future matching. An explanation of the work undertaken and included in this prototype application is in the section entitled

4. WP4: Prototype Demonstrator.

Publications:

1. Khan, S., Parkinson, S., & Murphy, C. (2023). Context-based irregular activity detection in event logs for forensic investigations: An itemset mining approach. Expert Systems with Applications, 233, [120991]. https://doi.org/10.1016/j.eswa.2023.120991
2. Khan, S., Parkinson, S., Roopak, M. & Murphy, C. (2023) Benchmarking a Parameterised Approach to Generating and Detecting Event Chains in Security Dataset, Submitted to Journal of Information Security and Applications

Project

As per the original plan, the project was broken down into 5 key work packages:

1. **Establishment and generation of key data sources (month 1):** This WP was to establish key information sources that can be extracted from a combination of commercial and civilian infrastructure. Information such as software used, event logs, etc. is available through the host operating system or security controls.
2. **Knowledge discovery, learning, and formulation (months 2-9):** Techniques that are capable of learning domain knowledge from the in-process monitoring of event-based data sources will be further extended to handle multi-vendor heterogeneous and incomplete data sources.
3. **Searching, realisation, and automated response (months 9-10):** This WP will develop mechanisms to utilise the acquired domain model, alongside monitoring systems to automatically suggest actions that can be executed based on matching preconditions.
4. **Prototype demonstrator (months 10-12):** The research deliverables have the potential to be used for many different purposes within and beyond the security sector.
5. **Dissemination, testing, and stakeholder engagement (months 1-12):** All surveys, technical developments, and findings will be disseminated with the project team, and subject to approval, have been published in peer-reviewed open-access journals.

In this project, the above plan was followed; however, the amount of effort required by each work package deviated from the planned commitment. The most notable change was that there became a strong and important need for a data generator to overcome the absence of event-based datasets with known event chains of varying characteristics. This resulted in WP1 becoming significantly longer; however, it resulted in project outcomes beyond those anticipated that will be highly beneficial in the cyber security research community. Another notable change was that the effort required for WP3 was reduced due to the ability to leverage prior research. The following subsections provide an overview of the work undertaken in each of the work packages and the main findings and contributions.

WP1: Establishment and Generation of Key Data Sources

In this work, a normalised event structure has been adopted to ensure that irrespective of vendor-specific structure, they are normalised and can be processed as part of one event-based system. This enables the combination of event datasets to identify event chains involving multiple systems. Further technical details of the approach are in the paper [2].

Normalised Structure

In terms of converting the event logs into a single dataset for mining, it has proven important to adopt a set-based presentation. This normalised structure ensures that any input dataset is converted into a representation that is consistent and allows the application of the techniques developed in this project. This involves taking different event datasets and placing them into a generic event structure, as defined below:

- D is used to model the set of event entries, where D = {E1, E2, ..., En}.
- The event, E = {T, ID, O, M}, where T is a timestamp, ID is a numeric event type ID, and O is the set of unique objects, such that O = {O_1, O_2, ..., O_N}, and M is a flag to label whether the event is part of a mitigative security action.
- Each entry, Ei, contains its relevant objects from O to represent an occurrence of the event.
- For example, D = {E1, E2, E3, E4}, where:
    - E1 = {4/10/2022 8:38:09,4567,{User1, Win7, Port:53176, NTLM}}

- E2 = {4/10/2022 8:39:04,1234,{User1, ReadEA, svchost.exe, IKE}}
- E3 = {4/10/2022 8:39:50,2345,{User2, ReadEA, System, NTLM}}
- E4 = {4/10/2022 8:10:10,5678,{User2, Win7, NtLmSsp, Winlogon}}

Where possible the 'ID' is taken directly from the event to establish type; however, with some events sources (e.g., Linux), an event type information is not provided. In this instance, we cluster events based on similar object contents to establish types. If a good clustering solution does not exist (e.g., not enough data or too varied), then no types are assumed, and each event is processed as unique. The timestamp is used to preserve temporal ordering and ensure that events of the same time and even contents can be differentiated.

Event Chains

The term 'event chains' is used to denote a sequence of events that are connected in terms of event information reported by a system that can be linked to security controls. For illustrative purposes, **Error! Reference source not found.** provides a high-level example of whereby a series of events detail malicious activity followed by a response. The Figure 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 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. This example is synthetic and for explanation purposes. It is used to graphically visualise an attack chain that can be detected through event log sources. Further, it demonstrates that the mitigative action can also be extracted and added to the event chain, going from encoding the detection to also including the mitigation.

In prior work, events relating to mitigative actions are labelled. In the Microsoft environment, where event types are pre-defined, it is possible to separate those presenting information based on system interaction and use with those presenting information resulting from a configuration change and defensive security behaviour (e.g., firewall rule change, antivirus quarantine). The labelling was performed based on event-type summary information provided by Microsoft[1].

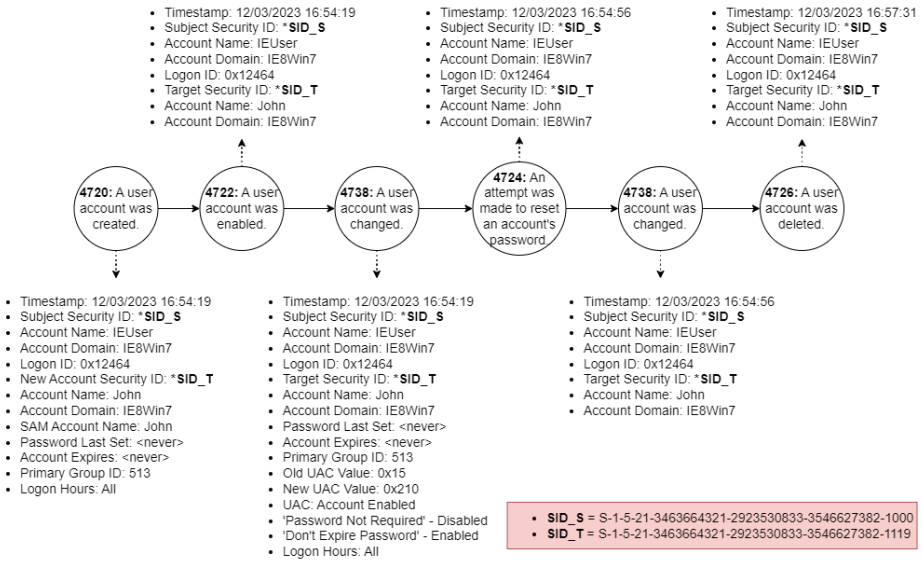


FIGURE 2 - EXAMPLE EVENT CHAIN

Parameterised data generation

After extensive research of available approaches (details in paper [2]), it was established that there is a strong need to develop a technique capable of generating synthetic data where ground truth knowledge is available, i.e., a thoroughly labelled dataset. Figure 3 presents the approach, which is presented in detail in the paper [2]. In summary, the event generation process consists of three main end-user functions (1) event chain generation, (2) noise generation, and (3) using a dataset alongside synthetically created event chains. Before these three functions can be used, there is a need to create a set of synthetic objects that represent the key-value pairs used in the event log entries. The assumption behind adopting an 'object' viewpoint here

[1] https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/plan/appendix-l--events-to-monitor

is that as previously mentioned, events often constitute key pieces of information, either in key-value form or value form. For example, Microsoft systems use a key-value form and Linux syslog uses only the object.

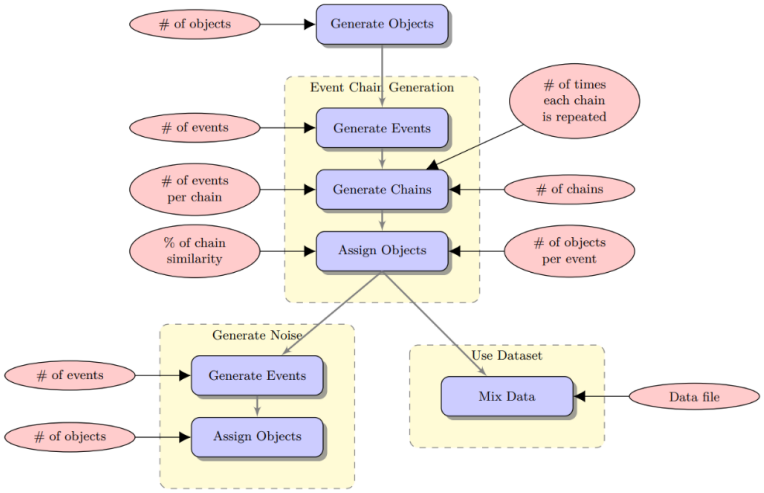


FIGURE 3 - SYNTHETIC EVENT GENERATION PROCESS

In terms of the three main end-user functions:

1. **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 that the events in an event chain share a relatively high number of common objects in comparison to events that are not connected. 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.
2. **Noise generation**: In addition to the chains generated for 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*.
3. **Use Dataset**: 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, 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.

WP2: Knowledge Discovery, Learning, and Formulation

Event chains are a sequence of events grouped 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. In this project, we developed an approach capable of detecting event chains. To evaluate how well our approach works, we used data produced from our data generator.

Unsupervised Clustering to Discover Event-Chains.

Our experimental research identified that an unsupervised clustering-based approach is suitable. The full details of the algorithms and why they were considered are available in the paper [2]. In summary, after trial-and-error testing, the algorithms of DBSCAN, OPTICS, Affinity Propagation, Mean Shift and Agglomerative. In the initial experimentation, we were interested in understanding the performance of the algorithms in terms of the time taken to perform the clustering and how well they identified the event chains. To measure their ability to identify event chains, we use the Adjusted Rand Index (ARI) score, which is a standardised measure as to how well two groups match, which in this research is the original dataset grouping and the

unsupervised algorithm's clustering. An ARI value of 1 means a perfect match and 0 is no match. As seen in Figure 4 DBSCAN outperforms in terms of time required and ARI.

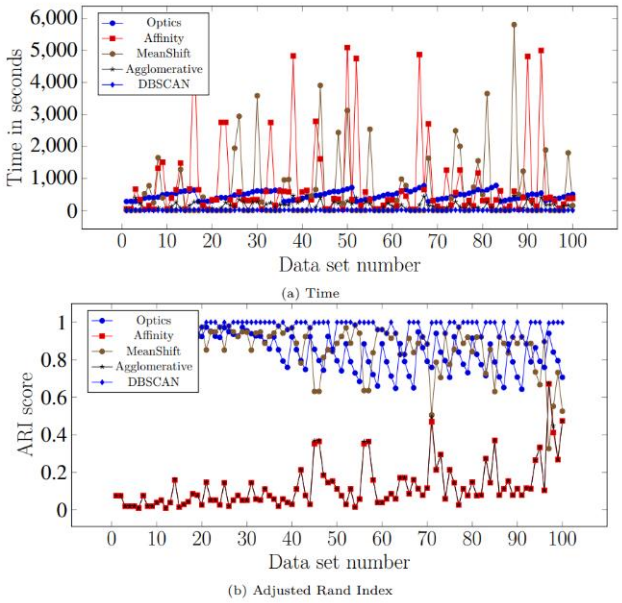


FIGURE 4 - PERFORMANCE RESULTS FROM 5 SELECTED CLUSTERING ALGORITHMS USING A SMALL DATASET

After identifying DBSCAN as the best, an exhaustive simulation was performed where the different permutations of event generator inputs (WP1) were considered to evaluate performance and capability. In the simulation, we considered event sizes ranging from 10,000-40,000 and between 10-40 chains, with varying lengths, similarity, and repetition. In total, the benchmark simulation considered over 12,000 event datasets with different characteristics. We were then able to analyse the results and establish which of the input parameters had the greatest impact on the ability to correctly identify event chains. As seen in Figure 5 and in descending order, the number of chain repetitions, total number of events, number of repetitions and number of events in each chain are identified as the most important.

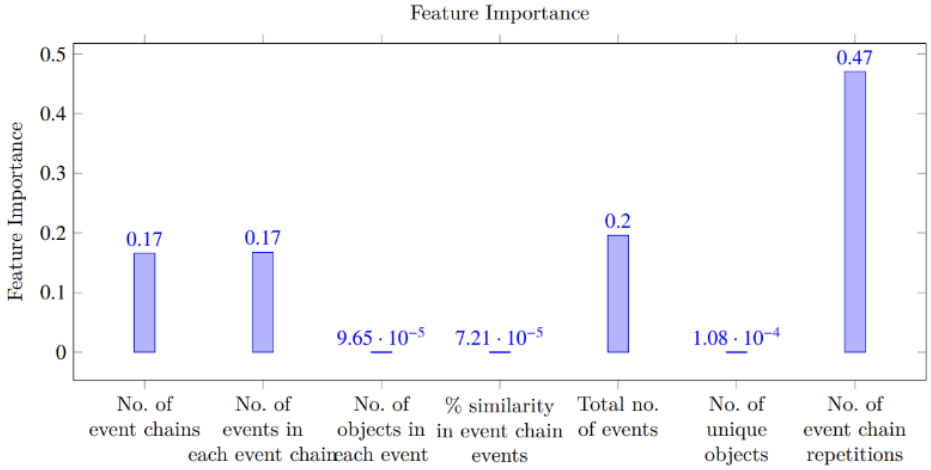


FIGURE 5 -FEATURE IMPORTANCE SCORE OF ALL SYNTHETIC EVENT GENERATION PARAMETERS

The outcome of this research is that the DBSCAN algorithm can be successfully used to identify event chains. The research also established the properties of event datasets that both increase and decrease detection rates. Furthermore, the benchmark analysis included in the paper [2] provides a robust benchmark for further research.

Mining Irregular Event-Chains

In addition to exploring the use of clustering algorithms to detect action chains, we also investigated the use of Association Rule Mining (ARM) techniques to detect irregular event chains, i.e., event chains that occur in low frequency. The motivation for exploring this work was due to previous work we had undertaken before

this project where ARM had demonstrated good capability in detecting frequently occurring event chains [3]. The research, as published in the paper [1], demonstrates that the approach can detect irregular event chains with an accuracy between 75-90%. Although this demonstrated reasonably promising results, it is below the capabilities of DBSCAN. Furthermore, ARM algorithms are iterative and require more processing time when compared to efficient clustering algorithms. For example, in the paper [1], the processing time required ranges from 5 to 40 minutes.

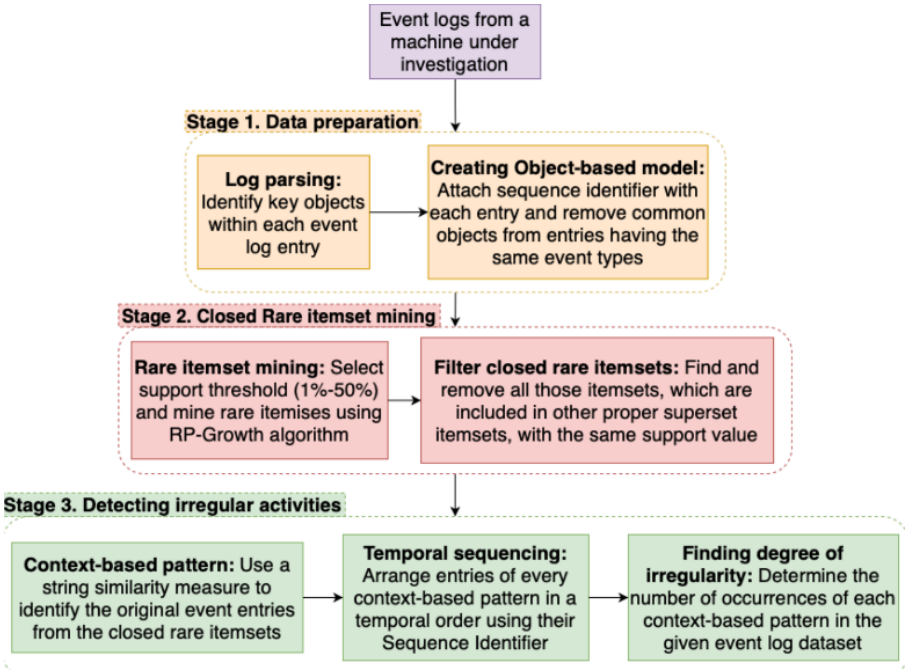


FIGURE 6 - IRREGULAR ASSOCIATION RULE MINING APPROACH

Figure 6 illustrates the approach taken from the paper [1], and the following list provides a brief explanation of each of the main phases:

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 event chains (also called *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 event chains.
3. **Detecting Irregular Activities**: Using the closed rare event chains to discover sequences or patterns of related event entries, which were triggered due to system activity, along with prioritising the patterns based on their level of rarity.

WP3: Searching, Realisation, and Automated Response

Once event-chains are detected, they can then be used when monitoring future event datasets for matching purposes. However, as mentioned in the introduction, in this work we recognise that an event chain with strong co-occurrence with contents will be detailing a specific system activity. For example, as shown in Figure 2, the first five events detail system interaction on an account level, before a change is reported in the sixth event to block the account. Therefore, we can state that the first 5 actions are the precondition i.e., what happens before a change is made, and event 6 is reporting the effect i.e., what is reported once a change is made. In this report, we leverage approaches from the model-based artificial intelligence community whereby *actions* are modelled to store knowledge on discrete activities that can be performed in specific domains. For example, in a logistics environment, actions exist to model the movement of goods between locations. In this work, we use an action to model the change in a system's security controls.

Action-based models are widely used in the Automated Planning (AP) community to deliberate over problems with a known set of actions to achieve a desired goal. In this project, we adopted the use of the Planning Domain Definition Language (PDDL) which is commonly used for encoding domain knowledge [4]. Given the knowledge is encoded into the PDDL domain action model and a problem file, various planners

are available based on general problem-solving techniques. They can generate plans concerning quality and time-based constraints. The process of generating domain models requires additional expertise in modelling complex domain actions. his has resulted in a pursuit of autonomously processing available data sources to learn domain knowledge. Based on prior research [3], The Local Search for Planning Graphs (LPG) [5] planner is used due to its good performance and support for PDDL.

To convert an event chain, human input is required to separate which events are related to the precondition, and which are related to the effect. This section presents the process of encoding temporal-association rules into an action-based domain model using the PDDL, without any human intervention. PDDL is a standardised format supported by numerous AP applications. A PDDL domain model consists of types of objects, predicates modelling facts, and functions to handle numeric functions and actions. A single action models an evaluation or configuration step to increase security. A precondition represents at what point the action is selectable, and an effect models a change to the objects, predicates, and functions. The first events selected by the analyst become the precondition, whilst the second set becomes the effect part of an action. The combined list of object names constitutes parameters for the action. Each domain action having one or more parameters will provide all `entities' that are required to make any configuration change.

Table 1 presents seven events that have been identified to be part of the same chain. It should be noted that this is a synthetic event chain, and hence the object names and values have numeric names. By taking the first three actions in the last as the precondition and the last four as the effect, the PDDL encoding shown in Figure 7 is created. The PDDL encoding models an action containing all unique objects. The objects are encoded in the precondition and effect. For example, '?ObjectValue24' is included in the precondition and effect. This representation enables the action to be matched on even event-based data sources where the correctly typed objects are all present. This presence allows an AP algorithm, such as LPG, to propose the execution of the action, modelling its effect on the underlying object-based model.

ID	Time Stamp	Key-Value pairs			
9228	10/10/2023 12:41	ObjectName11: ObjectValue11	ObjectName38: ObjectValue9	ObjectName8: ObjectValue16	ObjectName22: ObjectValue18
3054	10/10/2023 12:44	ObjectName35: ObjectValue24	ObjectName36: ObjectValue11	ObjectName14 :ObjectValue3 1	ObjectName15: ObjectValue25
3413	10/10/2023 12:53	ObjectName27: ObjectValue2	ObjectName26: ObjectValue32	ObjectName8: ObjectValue38	ObjectName9:O bjectValue33
2327	10/10/2023 12:55	ObjectName28: ObjectValue26	ObjectName14: ObjectValue27	ObjectName22 :ObjectValue9	ObjectName6:O bjectValue19
7068	10/10/2023 13:01	ObjectName32: ObjectValue9	ObjectName31: ObjectValue1	ObjectName11 :ObjectValue3 5	ObjectName12: ObjectValue39
3309	10/10/2023 13:02	ObjectName33: ObjectValue24	ObjectName16: ObjectValue26	ObjectName29 :ObjectValue1	ObjectName15: ObjectValue5
8314	10/10/2023 13:17	ObjectName34: ObjectValue24	ObjectName22: ObjectValue5	ObjectName1: ObjectValue3	ObjectName33: ObjectValue26

TABLE 1 - EVENT CHAIN IN TABULAR FORM

```
(:action action_9228_3054_3413__2327_7068_3309_8314
        :parameters ( ?ObjectValue11 - eo ?ObjectValue9 - eo ?ObjectValue16 - eo ?ObjectValue18 - eo
        ?ObjectValue24 - eo ?ObjectValue31 - eo ?ObjectValue25 - eo ?ObjectValue2 - eo ?ObjectValue32
        - eo ?ObjectValue38 - eo ?ObjectValue33 - eo ?ObjectValue26 - eo ?ObjectValue27 - eo
        ?ObjectValue19 - eo ?ObjectValue1 - eo ?ObjectValue35 - eo ?ObjectValue39 - eo ?ObjectValue5 -
        eo ?ObjectValue3)
:precondition
        (and
        (ObjectName11 ?ObjectValue11) (ObjectName38 ?ObjectValue9) (ObjectName8 ?ObjectValue16)
```

FIGURE 7 - EVENT CHAIN MODELLED IN PDDL

WP4: Prototype Demonstrator

A software prototype has been created to link together the key software deliverables resulting from the research and development work undertaken in this project. That is specifically (1) the event generator, (2) the clustering approach to identify event chains, and (3) the ability to convert event chains into actions. The prototype application has been written in C#.

Figure 8 provides a graphical illustration of the software components included in the prototype application. In terms of the 'Data Generator', the approach developed in WP1 is included. For 'Event Chain Learning', a C# implementation of DBSCAN that is comparable with the Python version used in WP2 is included. The final stage, 'Planning', includes the use of the LPG algorithm. An example of how to use the prototype is included in Appendix 1 – Prototype User Guide.

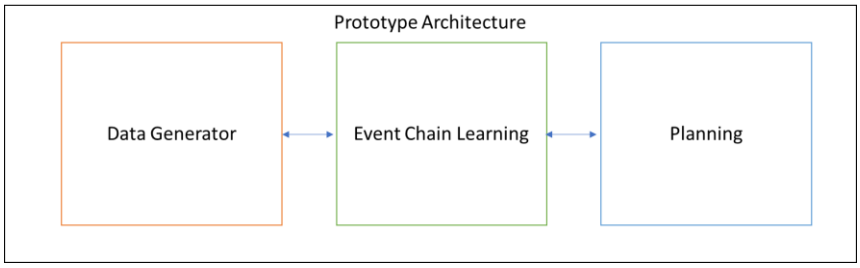


FIGURE 8 - PROTOTYPE ARCHITECTURE

WP5: Dissemination, testing, and stakeholder engagement

The two publications (papers [1]. [2]) have been created and disseminated during this project. Paper [1] has been accepted and published. Paper [2] is currently under review after receiving feedback from the first submission. Throughout the project, monthly meetings were held and the presentations, minutes and any relevant attachments were circulated by email. The project team attended the ARCD Show & Tell event at Cody Technology Park on the 25th of January 2023.

Conclusion

In this project, a novel event dataset generator has been produced to respond to the need for a parameterised approach to generate event datasets for event-chain mining tasks. The need has arisen through the absence of publicly available datasets with ground-truth knowledge of their event chains. The project has investigated and developed an approach to successfully identify event chains using an unsupervised clustering approach. It is then demonstrated how the event chains can be translated into PDDL and used alongside model-based reasoning techniques to identify and plan actions where there is a partial event-chain match in an event dataset matching that of the created action. The project overall has been successful, meeting its key aims. The project required more effort than expected to create the data generator and benchmark unsupervised clustering approaches. However, this was not detrimental to the project and all other objectives were met. The deliverables of this project provide a basis for future research into mining event chains, modelling and exploiting actions. The project team aim to acquire follow-on funding to pursue further avenues of research, such as the use of large language models.

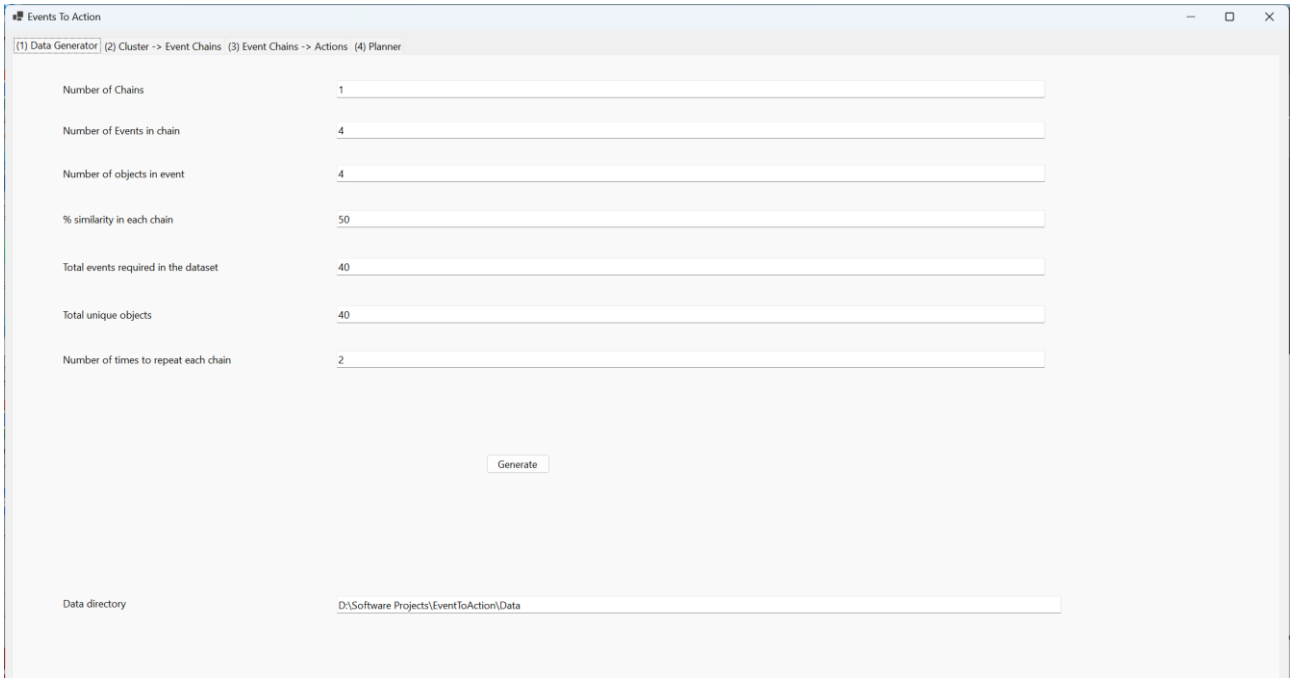
References

1. Khan, S., Parkinson, S., & Murphy, C. (2023). Context-based irregular activity detection in event logs for forensic investigations: An itemset mining approach. Expert Systems with Applications, 233, [120991]. https://doi.org/10.1016/j.eswa.2023.120991
2. Khan, S., Parkinson, S., Roopak, M. & Murphy, C. (2023) Benchmarking a Parameterised Approach to Generating and Detecting Event Chains in Security Dataset, Submitted to Journal of Information Security and Applications
3. Khan, S., & Parkinson, S. (2019). Discovering and utilising expert knowledge from security event logs. Journal of Information Security and Applications, 48, 102375.
4. Haslum, P., Lipovetzky, N., Magazzeni, D., Muise, C., Brachman, R., Rossi, F., & Stone, P. (2019). *An introduction to the planning domain definition language* (Vol. 13). San Rafael, California: Morgan & Claypool.
5. Gerevini, A., Saetti, A., & Serina, I. (2003). Planning through stochastic local search and temporal action graphs in LPG. *Journal of Artificial Intelligence Research*, *20*, 239-290.

Appendix 1 – Prototype User Guide

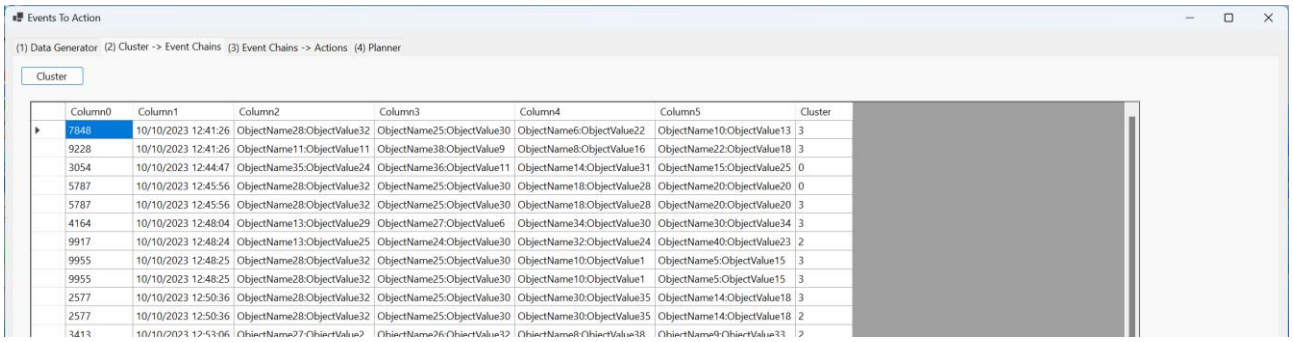
The prototype application has a simplistic graphical user interface to guide the user through the different phrases which are on separate tabs. This includes (1) Data Generation, (2) Cluster -> Event Chain, (3) Event Chains -> Actions, (4) Planner. The following subsections explain each of the tabs and what functionality is available to the user.

Dataset Generation



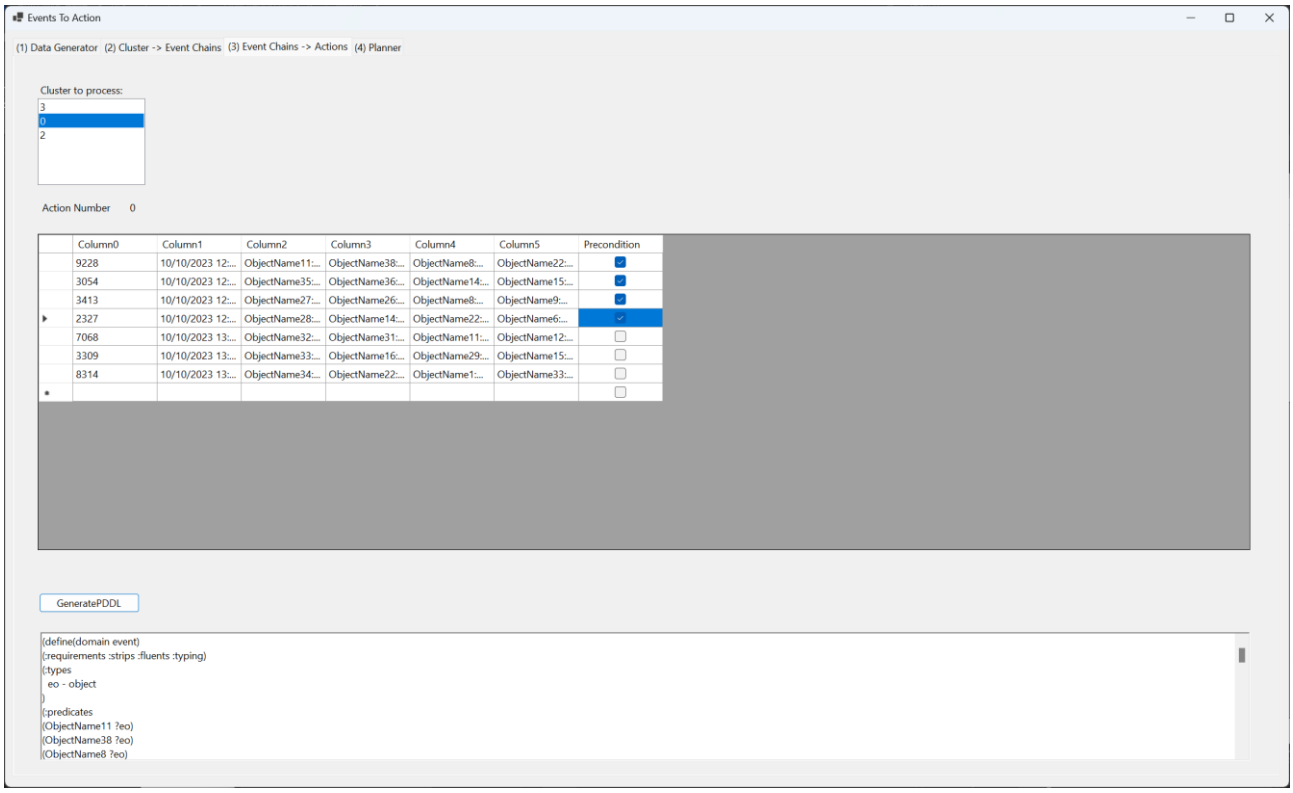
The Dataset Generation tab provides a graphical front-end to provide the parameters and also provides the 'data directory', which is where the output files will be stored. The output files are named using the parameters, so you only need to 'Generate' the dataset if you do not already have a file of that specification in the data directory. The individual parameters are self-explanatory; however, those interested in understanding more information on how they are used are directed to paper [2].

Cluster -> Event Chains



On the 'Cluster -> Event Chains' tab, you will need to first press 'Cluster' which will run the C# implementation of DBscan on the dataset generated or specified on the Data Generation tab. The above screenshot shows how the result is shown. This screen shows the events and their objects in a tabular form, with a 'Cluster' column added at the end. The number in this column represents which cluster each event is part of, and if you choose, you can click on the cluster column to organise the data more appropriately.

Event Chains -> Actions



This tab allows you as the user to inspect each of the ‘Cluster’ (event-chain) by selecting one from the ‘Cluster to process’ list, which will populate the tabular view with all unique events. You can then select which of the events should be regarded as the ‘Precondition’. All the ones not selected are as default assumed to be the effect. The ‘GeneratePDDL’ button will process the assignment you have made for each of the clusters and generate the PDDL. The PDDL is shown on screen in the text window, for research and development. The PDDL files are also output in the directory specified on the ‘Data Generation’ tab in a ‘planner’ subdirectory. These files are what will be used on the ‘Planner’ tab.

Planner

Finally, the ‘Planner’ tab allows you to use the PDDL action model created on the ‘Event Chains -> Actions’ tab. By pressing ‘Planner’ the PDDL files will be used to determine if any of the actions can be utilised on the event dataset that was generated on the Data Generator tab. If any actions are identified, they will be listed in the list view as shown above, where two actions have been identified.