ThreatTrace: Cyber-Attack Detection through Trace Abstraction and Soft Clustering

Andrzej Janusz[1,2][0000−0002−9763−1399], Savandi Kalukapuge[1][0009−0007−6102−3440], Moe Thandar Wynn[1][0000−0002−7205−8821], and Arthur H.M. ter Hofstede[1][0000−0002−2730−0201]

[1] School of Information Systems, Queensland University of Technology, Australia
[2] Institute of Informatics, University of Warsaw, Poland
{andrzej.janusz, s.kalukapuge, m.wynn, a.terhofstede}@qut.edu.au

Abstract. The growing sophistication of cyber threats has raised the importance of providing a proper level of cybersecurity to networks and interconnected devices. Process mining provides methods to analyse sequences of activities performed by operators or programs on such devices to detect cyber-attacks from a novel perspective, i.e., as continuous processes composed of observable system events. However, processes like that, especially in domains such as the cybersecurity of the Internet of Things (IoT) or network devices, are usually characterised by large variability and complexity, which make them difficult to model. We propose a method called ThreatTrace for simplifying and creating embeddings of complex process traces that can often be observed in event logs from such domains. The novelty of ThreatTrace stems from integrating process mining techniques into the cyber-threats discovery workflow. It combines process trace abstraction, embedding, and soft clustering to generate compact process variant representations that preserve information about patterns of interest, e.g., traces of potential cyber-attacks. The use of compacting patterns, which can either be automatically extracted from data or manually designed by experts, not only allows us to reduce the irrelevant variation in observed process traces but also provides an opportunity to evaluate and tune the constructed trace embeddings. Then, soft clustering of process variants is performed to determine cluster membership values of individual traces. Our experiments show that such information extracted from event logs improves the detection of cyber-attacks over approaches that do not consider the process perspective, particularly in the context of IoT device and network cybersecurity.

Keywords: Process Mining · Cybersecurity · Trace Abstraction · Process Trace Embeddings · Soft Clustering of Traces.

1 Introduction

The cybersecurity landscape is increasingly critical due to our reliance on connected technology. Process mining [1] offers a fresh perspective by analysing sequences of actions performed by users or system programs as distinct process

instances. This approach helps to uncover hidden patterns and detect deviations that may signal malicious activity. In an ideal scenario, network devices, particularly IoT devices, exhibit regular behaviour. When a deviating activity is observed, it may be a potential indicator of cyber-attacks. In practice, however, cybersecurity-related processes are challenging to analyse due to their inherent complexity and variability [15]. Process mining methods can empower cybersecurity analysts to trace and better understand attack paths, enabling them to undertake proactive cybersecurity measures.

Understanding cyber-attacks as processes allows for a more nuanced comprehension of the intricate sequences of events that lead to security breaches. However, a significant challenge in viewing a cyber-attack as a process lies in identifying the appropriate level of abstraction. Without proper aggregation, cyber-attacks that share similar characteristics may be treated as disparate incidents, hindering effective analysis and detection efforts. In real-world systems, the presence of multiple concurrent processes further compounds this challenge. While some processes may exhibit indicators of malicious activity, others may operate entirely within the bounds of normal behaviour. The ability to detect cyber-attacks with the shortest possible lags, i.e., in near real-time, amidst the noise of routine operations in a system is essential.

In this research, we aim to address the above-mentioned challenges. We propose a novel method, called ThreatTrace, for the analysis of cybersecurity event data that allows accurate detection of cyber-attacks. ThreatTrace combines a frequent pattern-based trace abstraction technique with embedding learning and soft clustering. Although various techniques in the cybersecurity domain use rules and patterns to identify possible cyber attacks [11], there has been a limited focus on using trace abstraction methods from the field of process mining to address this challenge. The novelty of our approach stems from the use of a new stochastic model in which traces are represented by multivariate Gaussian distributions defined in the activity embedding space. It can be considered as a process trace abstraction technique whose goal is to create a concise representation of event sequences in cybersecurity logs. We show that traces in this representation can be clustered using standard methods. We apply the fuzzy c-means (FCM) algorithm to extract trace-cluster membership values and use them as higher-level features. We then train machine learning (ML) models to detect cyber-attacks in newly observed logs. Our experiments show that using information extracted from event logs improves the detection of cyber-attacks over approaches that do not consider the process perspective.

In the following Section 2, we discuss related work. Then, in Section 3, we explain the major steps in our workflow, focusing on how ThreatTrace abstracts process traces and extracts higher-level features for the detection of cyber threats. In Section 4, we describe datasets, which we use to evaluate our approach and discuss the results of our experiments. Then, in Section 5, we reflect on the main limitations of ThreatTrace and point to possible directions for future research. Lastly, Section 6 concludes the paper.

2 Related Work

Process mining has emerged as a valuable tool in cybersecurity for analysing activity sequences and uncovering deviations indicative of malicious behaviour. As argued by van der Aalst [1], process mining offers inherent explainability over traditional black-box machine learning models through its process-oriented approach. For example, Macák et al. [15] reviewed its application in domains such as smart grids, network systems, and IoT devices, highlighting its role in visual analysis and conformance checking. They also demonstrated how this process-oriented view benefits cybersecurity applications by enabling interpretable threat detection. Zhong et al. [28] applied process mining techniques to network packet data preprocessing for intrusion detection, demonstrating effectiveness against common attack patterns using real-world datasets such as CSE-CIC-IDS2018.

Van Zelst et al. [27] proposed event abstraction taxonomies to simplify event logs while preserving meaningful security patterns, mitigating the "spaghetti effect" of complex process models. While their approach offers a foundation for log simplification, it lacks the integration of embedding learning and clustering techniques that facilitate handling the dynamic nature of cyber-attacks. Techniques like frequent sequence mining [26] and context-aware trace clustering [5] have advanced automation in process trace abstraction. However, unlike in our approach, they were applied separately and did not take advantage of the possible synergy between them.

Clustering methods are effective in processing large-scale, variable cybersecurity data. Song et al. [19] used hierarchical clustering with sequence alignment to enhance trace clustering efficiency, while agglomerative techniques like Ward's method [3] have improved data aggregation in process mining. While hard clustering assigns each data point to exactly one cluster, soft clustering allows data points to belong to multiple clusters simultaneously with different degrees of membership [4]. This distinction is crucial, as hard clustering may fail to capture overlaps between normal and malicious activities. Soft clustering, as highlighted by Xu and Wunsch [25], addresses this, which makes it viable for detecting evolving attack patterns in cybersecurity.

Activity and trace embedding techniques have emerged as powerful tools for representing process traces. NLP-inspired methods like trace2vec [14] have been used in several process mining tasks, such as trace clustering. However, they struggle with generalising to unseen trace variants – a common challenge in dynamic environments like IoT networks, limiting their applicability. Advancements in RNN-based representations [10] and transformer-based embeddings [21] capture sequence patterns for anomaly detection but are computationally intensive and require extensive labelled data, which is often impractical. GloVe [18] offers a robust alternative for generating activity-level embeddings that can be aggregated to represent traces. It enables real-time processing of new traces without retraining. Studies have shown that GloVe consistently outperforms word2vec-based approaches in various NLP tasks such as word analogies and text classification [18, 23].

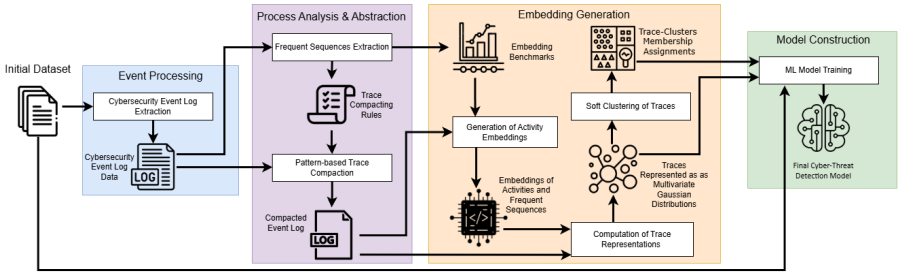

Fig. 1. A high-level overview of the ThreatTrace workflow.

ThreatTrace integrates these various approaches, i.e., process trace abstraction, learning activity embeddings, and soft clustering, to tackle the challenges of cyber-attack detection in IoT and network environments. Its distinct characteristic is the novel stochastic model of process traces. In this model, traces are represented as multivariate Gaussian distributions in the activity embedding space. We show that such representation not only enhances cyber-threat detection accuracy but is also computationally efficient and can be used for monitoring complex processes in near-real time.

3 Constructing Abstractions of Traces in ThreatTrace

When dealing with real-world processes in complex domains such as cybersecurity, standard approaches to process modelling may face the problem of large variability in the observed process traces. As a result, constructed process models can become overly large and difficult for domain experts to analyse. ThreatTrace addresses this issue by constructing abstractions of process traces and concisely representing them as multivariate Gaussian distributions over the activity embedding space. Figure 1 illustrates the ThreatTrace workflow. It consists of four main phases: event processing, process analysis and abstraction, and embedding generation, followed by model construction. The workflow begins with event log extraction from the initial dataset. In the process analysis and abstraction phase, frequent sequences are extracted and used to create trace compacting rules, which are then applied to simplify the event logs. The embedding generation phase is the core of ThreatTrace. Activity embeddings are generated and evaluated using automatically created benchmarks. These embeddings, along with embeddings of frequent sequences, are used to construct trace representations, i.e., The traces are modelled as multivariate Gaussian distributions, defined in the space of activity embeddings and processed through soft clustering to determine cluster memberships. Finally, in the model construction phase, these derived features are used to train ML models for cyber-attack detection.

3.1 Compacting of process variants

Our overall goal for this step is to reduce the number of variants that are considered different even though they are essentially the same from a cyber-attack point of view while preserving outliers that may be indicators of potential attacks. Our secondary goal is to capture the most substantial examples of the *directly follows* relation that we want to reflect in constructed embeddings.

Let us denote the j-th process trace in available data by a sequence of action symbols $T^j = [X_1^j, X_2^j, \ldots, X_{N_j}^j] \in T$, where each X_i^j is a symbol corresponding to one of the possible actions from the set A, i.e., $X_i^j \in A = \{A_1, \ldots, A_n\}$. Let us also assume that we have an ordered list of compacting rules $R = [R_1, \ldots, R_m]$, where each rule R_i indicates a substitution of a subsequence $[Y_1, Y_2]$ with a new symbol $A'_k \in A'$. For simplicity, we will denote such a rule by an implication $[Y_1, Y_2] \rightarrow A'_k$. It is worth noting that symbols $Y_1, Y_2 \in A \cup A'$, and $A \cap A' = \emptyset$.

The original process traces from the available data can be compacted by recursively applying rules R_i to each trace $T^j \in T$ in the order indicated by R. For example, if we consider a list of rules $R = [R_1, R_2, R_3]$, such that:

$$R_1 = ([a, b] \rightarrow b'),\ R_2 = ([a, a] \rightarrow a'),\ R_3 = ([a', a] \rightarrow a'),$$

then a process trace

$$t = [a, a, b, a, a, a, b, c, a, a, a]$$

would be compacted to:

$$t_c = [a, b', a', b', c, a'] \ .$$

After the application of R_1 trace t would be reduced to: $[a, b', a, a, b', c, a, a, a]$. Then, the application of R_2 would further reduce it to: $[a, b', a', b', c, a', a]$, and the final sequence of symbols would be obtained by the application of R_3.

Rules from R can be automatically extracted from data using an arbitrary frequent sequence mining algorithm, such as SPADE [26], or they could be provided by domain experts. It is typically beneficial to sort the list R decreasingly by rule support. Rules that are closer to the beginning of the list are likely to contribute more to the reduction of process traces. However, more advanced optimisation heuristics, such as the genetic algorithm [22], can also be used to find the ordering of compacting rules that yields the highest trace compression ratio.

3.2 Computation of action and trace embeddings

Trace embedding techniques have been used by other researchers to facilitate the clustering of process traces [14]. One major limitation of the *trace2vec* technique used in that study is that it is not able to infer the representation of new, previously unobserved trace variants. Traces like that are common in practice, especially in domains such as cybersecurity, where the number of different possible activities can be large. To bypass this limitation, we propose to compute the embeddings on the activity level and represent traces as aggregations – clusters modelled by Gaussian distributions of activity embeddings. In this way, ThreatTrace can embed newly observed traces as long as the set of possible activities

remains fixed. Moreover, we can apply this approach in an event streaming setup to facilitate cyber-threat detection in near real-time.

Another major difference with the *act2vec* and *trace2vec* methods proposed in [14] is that the embedding learning algorithm used in ThreatTrace is based on GloVe [18]. Several studies demonstrated that Glove consistently outperforms *word2vec* used in *act2vec* and *trace2vec* in various NLP tasks, such as word analogies, prediction of synonyms, NER, and text classification [18,23]. We train it on event log data by considering consecutive action codes in each trace. To guide the embedding learning process, we add special tokens to the list of activities, which indicate whether the trace was observed during the cyber-attack. We use standard settings for the context window size and other GloVe parameters. However, we tune the embedding size using a set of benchmark similarity queries that we derive from the frequent patterns discovered during the initial trace compacting.

The construction of benchmark queries is motivated by the word similarity task, commonly used as an evaluation task for natural language word embeddings. A desired feature for word embeddings is their ability to represent semantic patterns as linear translations [17]. For instance, we would like to see that vec("Warsaw") - vec("Poland") + vec("Australia") is more similar to vec("Canberra") than to any other word embeddings, especially for words representing names of other capital cities. Since the compacting patterns in ThreatTrace are sequences of individual activities, and the embeddings are computed for both the individual activities and their sequences, we can construct and evaluate similar algebraic expressions. For example, if we have the embeddings for $\{a, b, c, [a, b], [a, c]\} \subseteq A \cup A'$, we would expect that

$$vec([a, b]) - vec(b) + vec(c) \simeq vec([a, c]),$$

where $\simeq$ denotes the similarity relation. We extract multiple benchmark similarity queries using templates like the one above. We compute the average similarity and rank of expected outcomes and use these two indicators to select the optimal vector size for the activity embeddings.

Finally, having the embeddings for activities and activity sequences, we represent traces as clusters of the corresponding embedding vectors. We model these clusters as multivariate Gaussian distributions. Trace embeddings are the descriptions of such clusters, i.e., for each trace, ThreatTrace stores the centroid and a covariance matrix of the distribution. Note that it is often impractical to store the whole covariance matrix. For the sake of conducted experiments, we only consider its diagonal, i.e., the vector of variances in each dimension. Thus, the vector representation for a trace T^j in ThreatTrace can be denoted as:

$$vec(T^j) = (\mu_1^j, \ldots, \mu_k^j, \sigma_1^j, \ldots, \sigma_k^j) \tag{1}$$

where μ_i^j is the average of the i-th dimension of activity embeddings from the trace T^j, i.e.,

$$\mu_i^j = \frac{1}{N_j} \sum_{l=1}^{N_j} vec_i(X_l^j) \tag{2}$$

and σ_i^j is the standard deviation of the i-th dimension of activity embeddings from the trace T^j, i.e.,

$$\sigma_i^j = \sqrt{\frac{1}{N_j - 1}\sum_{l=1}^{N_j}\left(vec_i(X_l^j) - \mu_i^j\right)^2} = \sqrt{\frac{1}{N_j - 1}\sum_{l=1}^{N_j}\left(vec_i(X_l)^2 - (\mu_i^j)^2\right)} \quad (3)$$

In the notation above, we denote the i-th dimension of the embedding for activity X_l as $vec_i(X_l)$. From the Formulas 2 and 3, we see that to facilitate the aggregation of activity embeddings and trace representations, instead of the vector (1), it is more practical to store the following vector of sums:

$$vec^*(T^j) = (N_j, \sum_{l=1}^{N_j} vec_1(X_l), \ldots, \sum_{l=1}^{N_j} vec_k(X_l), \sum_{l=1}^{N_j} vec_1(X_l)^2, \ldots, \sum_{l=1}^{N_j} vec_k(X_l)^2) \quad (4)$$

Such a representation is additive, making it trivial to combine two or more partial traces into a single trace representation. This characteristic is vital, e.g., when dealing with streams of system events. Additionally, the same representation can be used to represent whole trace clusters and facilitate the cluster analysis task.

3.3 Soft clustering of process variants

After the computation of the process trace representations, the next step in ThreatTrace is the cluster analysis of process variants. In our approach, individual process traces are modelled as Gaussian-distributed clusters of embeddings of activities and activity sequences. Thus, a natural metric for the computation of distances between the traces would be the Wasserstein distance [7]. If we denote the Wasserstein distance between two normal distributions by $d_W(N(\overline{\mu^1}, V^1), N(\overline{\mu^2}, V^2))$, were $\overline{\mu^j}$, are vectors of means, and V^j are the corresponding covariance matrices, then:

$$d_W(N(\overline{\mu^1}, V^1), N(\overline{\mu^2}, V^2)) = \sqrt{||\overline{\mu^1} - \overline{\mu^2}||_2^2 + ||\sqrt{V^1} - \sqrt{V^2}||_{Frobenius}^2} \quad (5)$$

Since in our representation of traces (Formula 1), we consider only the diagonal of the covariance matrix, the Wasserstein distance formula simplifies to:

$$\begin{aligned} d_W(T^1, T^2) &= \sqrt{||\overline{\mu^1} - \overline{\mu^2}||_2^2 + ||\overline{\sigma^1} - \overline{\sigma^2}||_2^2} \\ &= \sqrt{\sum_{i=1}^{k}(\mu_i^1 - \mu_i^2)^2 + \sum_{i=1}^{k}(\sigma_i^1 - \sigma_i^2)^2} \\ &= d_E\left(vec(T^1), vec(T^2)\right) \end{aligned} \quad (6)$$

where $d_E\left(vec(T^1), vec(T^2)\right)$ is the Euclidean distance between vector representations of the traces T^1 and T^2. Equation (6) allows us to cluster traces using the Euclidean distance and standard algorithms, such as *k-means*. However, we

propose a more refined approach. Instead of dividing traces into disjoint sets, we apply the *fuzzy c-means* (FCM) algorithm [12] and compute their membership degrees to identified clusters. FCM minimises the following objective function:

$$L(W,C) = \sum_{j=1}^{|T|}\sum_{k=1}^{|C|} w_{j,k}^{m}||vec(T^j) - c_k||^2, \qquad w_{j,k} = \frac{1}{\sum_{i=1}^{|C|}\left(\frac{||vec(T^j)-c_k||}{||vec(T^j)-c_i||}\right)^{\frac{2}{m-1}}} \tag{7}$$

where $W = w_{j,k} \in [0,1], j = 1,\ldots,|T|, k = 1,\ldots,|C|$ are cluster membership degrees for each trace and cluster, C is the set of identified cluster centroids, and $m \in (1,\infty)$ is a constant called the fuzziness degree. By using the online version of the FCM algorithm [4] in combination with our internal representation of process traces, we can easily apply ThreatTrace in the streaming setting for continuous monitoring of cyber threats.

4 Dataset Descriptions and Experimental Results

In our experiments, we consider two datasets from the cybersecurity domain. The first one is related to the detection of malicious activity in the operations of IoT devices. The second one considers the recognition and scoring of suspicious network activity. The datasets come from our industry partners and were previously used in international data science competitions organised at the KnowledgePit.ai web platform. We provide a processed version of those datasets, as well as the source code of conducted experiments in our GitHub repository[3].

4.1 Cyber-Attacks on IoT Devices

Our first case study uses a dataset collected from an IoT device, specifically a Raspberry Pi, as well as other devices that were responsible for generating and executing HTTP traffic and attacks [8]. It was created using an experimental environment developed in a project conducted by three companies, i.e., EFIGO, EMAG, and QED Software [2], focused on the cybersecurity of IoT devices. To obtain a sample that contained both regular and anomalous device operations, authors modelled the normal operation of the device and conducted cyber-attacks by exploiting some known system vulnerabilities. This involved the generation of typical network traffic and triggering standard system processes. It also required the manual design and automatic execution of scenarios for two different types of cyber-attacks on the device. More details about this experimental environment and the data generation process can be found in [8].

[3] The supplementary materials and source code are available at https://github.com/janusza/ThreatTrace-Cyber-Attack-Detection

The raw data was collected from the monitored devices as system audit logs. These logs were parsed and transformed into separate CSV tables corresponding to 60-second time windows (20 044 files in total). We refer to each such file as a single data case. The files were automatically labelled using information about the timings of the scheduled attacks. Different types of attacks were not distinguished – only the binary information about the presence of an attack was kept. The distribution of this binary target variable is highly imbalanced. Only 698 files were collected during cyber-attacks on the device (3.48%). Such a class imbalance is typical in the cybersecurity domain. This dataset was divided into two sets of files. Training data containing 15 027 CSV tables, and test data containing 5017 tables. The original data division in the competition was based on time, i.e., test tables corresponded to time windows strictly after the training data. We keep this division in all our experiments.

The rows of each CSV table correspond to individual system events recorded in the corresponding period. Events were described by 40 attributes, including information such as the event timestamp, process ID (PID), system call (action type), whether the system call was successful, and the process path. The attributes also included higher-level features extracted from the logs, such as counts of system kernel calls or the number of active services. We show an exemplary snippet of one of the CSV tables in Appendix A included in supplementary materials[3]. We focus on system calls associated with particular PIDs, i.e., we define an activity as a combination of the system call, user group, and the indicator of whether the system call was successful. We associate sequences of activities with the same PID in the 60-second period with a single instance (a case) of the investigated process. In this way, we have traces of multiple observed processes in each of the available CSV tables. The average number of process traces in a single table is 32.36, and the average number of events associated with a trace is 59.21. The total number of process traces in the training and test data is 648 539, and they are composed of 38 399 367 events. There are 77 different basic action types in the data and we extend this set to 193 activities through mining frequent sequences and applying compacting rules. Our experiments investigate the impact of information about the underlying processes, extracted using ThreatTrace, on the performance of cyber-threat detection models that were used as the baseline in the data science competition FedCSIS 2023 Challenge [8].

It is worth noting that the original dataset used in the challenge was flawed. The operating system on the monitored IoT device was never restarted during the data generation. As a result, the scheduled attacks were performed using programs that were given the same PIDs for every execution. Consequently, even though the training and test data corresponded to different time periods, the cyber-attacks could have been easily identified by the presence of specific PIDs in the audit logs from a given time window. Of course, such facilitation would be unrealistic in a real situation and can be considered as a severe data leak – in real life, each cyber-attack instance would be associated with a different set of randomly generated PIDs. Thus, in our experiments, we make independent PID assignments in every considered time window.

4.2 Prioritisation of Alerts for Suspicious Network Event

The second case study is based on data provided by the Security on Demand (SOD) company (currently DeepSeas) for IEEE BigData 2019 Cup [13]. The task in this competition was to distinguish between truly suspicious events and false alarms within the set of alerts derived from network traffic data that Security Operations Center (SOC) team members at SOD analyse on an everyday basis.

The dataset from this competition is still available at KnowledgePit.ai. It is composed of three tables. The first one contains 59 427 rows corresponding to so-called, *threatwatch alerts* investigated by the SOC team at SOD during the period between October 1, 2018, and March 31, 2019. Alerts in this table are described by 61 columns which represent information that is available to security operators and analysts during the investigation of suspicious events. Additionally, for each row, there was a label indicating whether the associated alert was considered "serious" by the SOD's analysts. Similarly to the first dataset, the target class in this dataset is highly imbalanced with only 5.76% positive cases. The competition organisers divided this data into training and test sets. The test set covered the alerts investigated in the last two months of the data collection period, and we kept this division in our experiments.

The second and third parts of the data are composed of two types of log files. We use them jointly in our experiments to extract features and learn representations of process traces. The second data part contains so-called *localized alert* logs, i.e., sequences of alerts related to individual *threatwatch alerts*. Events in this log correspond to alerts generated by expert system rules developed by the Threat Reconnaissance Unit at SOD [13]. In total, this data part contains 8 690 704 events described by a mixture of 20 numeric and symbolic features. The third part of the dataset contains a comprehensive event log of all network events registered by SOD's data collector devices within 24 hours, counting back from the corresponding *threatwatch alert's* timestamp. Such event logs can be extremely large - our dataset has 4 384 234 352 events described by 26, mostly symbolic, features (430GB of data). The complexity of this event log is also reflected in the large number of different activity types and the variability of process traces. Originally, there were 323 action types in the data and after mining frequent sequences and applying trace compacting rules, this set was extended to 772 action types.

4.3 Experimental Results

In this subsection, we present the results of experiments conducted to verify how the information extracted using ThreatTrace from event log data impacts the performance of cybersecurity threat detection models. We use datasets described in previous subsections and compare the quality of two prediction algorithms, i.e., the LASSO-regularised logistic regression (GLMNet) implemented in the *glmnet* library [20] and the decision tree gradient boosting model (XGBoost) implemented in the *xgboost* library [6]. While gradient boosting can be regarded as state-of-the-art in the tabular data classification task, logistic regression is

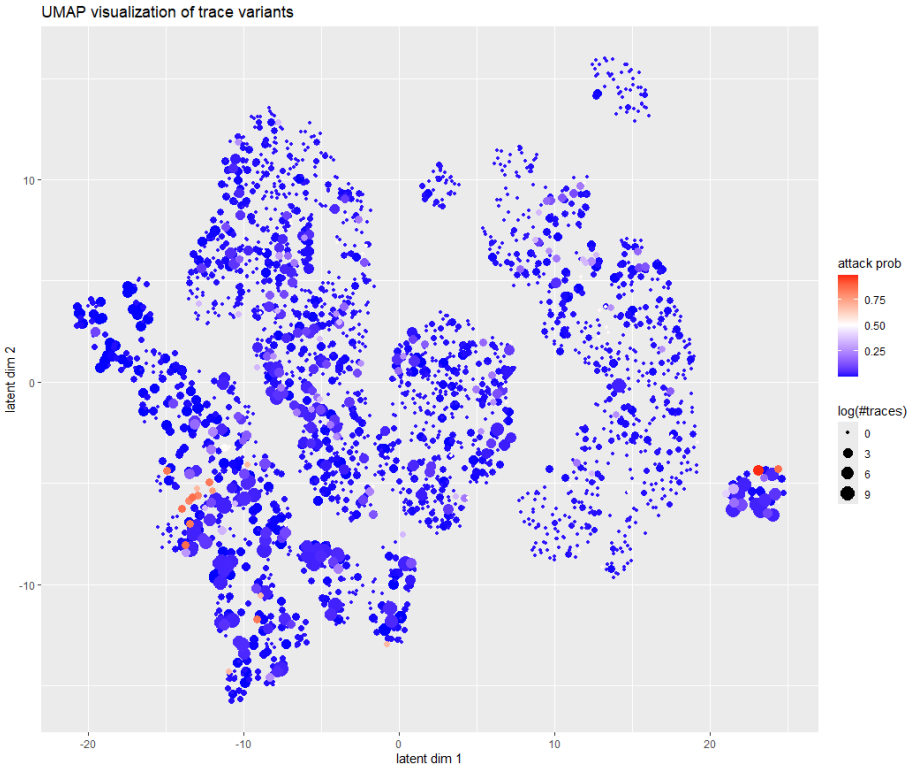

Fig. 2. UMAP visualisation of abstracted process variants from the FedCSIS 2023 Challenge dataset with a cyber-attack conditional probability heatmap.

still a common choice due to its high power efficiency. Moreover, the simplicity and low computational cost of inference make the logistic regression suitable for operating on mobile IoT devices. The baselines used in the comparison come from the data science competitions that are the source of our datasets [8,13]. We also added a reference model that is based on the feature extraction procedure inspired by descriptions from post-competition publications.

We trained the prediction models with the same hyperparameter setting for each of the compared representations. In the case of GLMNet, we set the *alpha* value to 1.0, i.e., only the LASSO regularisation was used. The *lambda* value was automatically tuned during the model fitting based on the results of 10-fold cross-validation on training data. The gradient boosting model used 2000 trees whose maximal depth was set to 10. The regularisation parameters *alpha* and *lambda* were set to 20.0 and 5.0, respectively. Additionally, to avoid overfitting, column and case sub-sampling were used with sampling rates of 0.8 and 0.95. These settings were chosen through the grid search for the representation that was obtained using a feature extraction approach inspired by descriptions of the corresponding competition baselines.

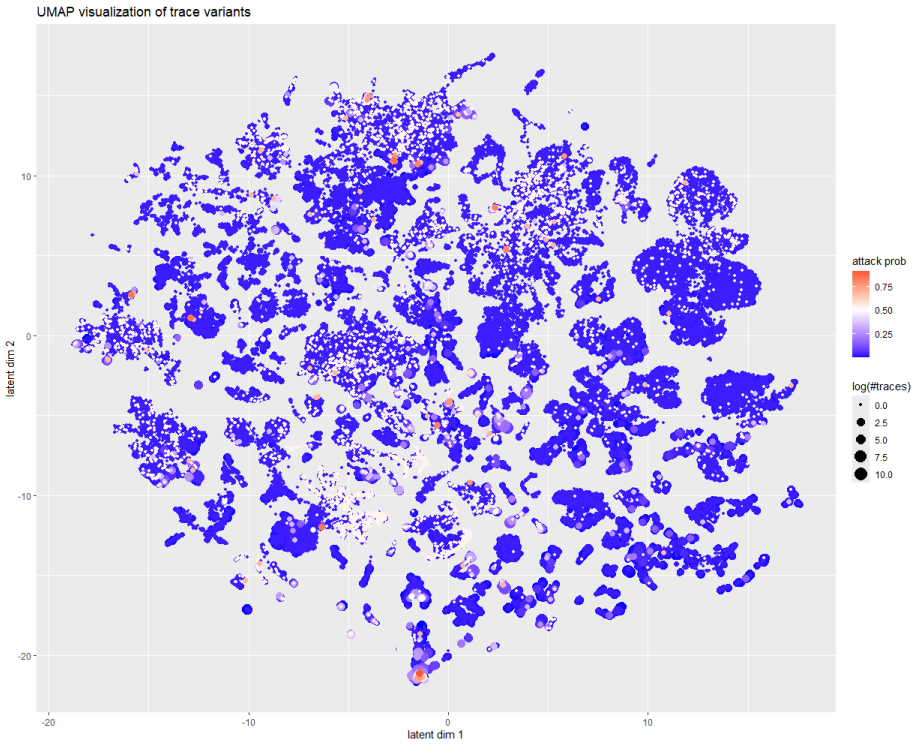

Fig. 3. UMAP visualisation of abstracted process variants from the IEEE BigData 2019 Cup dataset with a heatmap of the conditional probability of positive cases.

We tested ThreatTrace with three different embedding sizes. Their selection was made based on the results of the embedding evaluation procedure described in Subsection 3.2. The evaluation results suggested that increasing the dimensionality of activity embeddings is likely to improve their quality. However, it also brings diminishing returns. This observation is consistent with the final evaluation outcomes shown in Table 1. We visualised abstracted process traces from both datasets using the UMAP algorithm [16]. In Figures 2 and 3, the heat-map expresses the probability of the positive class conditioned on the presence of the corresponding abstracted trace variant. They were obtained using the 32-dimensional activity embeddings for the FedCSIS 2023 Challenge data and 40-dimensional embeddings for the IEEE BigData 2019 Cup data. We notice that variants associated with high cyber-attack probabilities are usually located in the boundary regions of local variant clusters. We also notice that for both datasets the initial stage of trace abstraction using the compacting rules significantly reduced the overall number of trace variants. For the first dataset, the reduction was by 21.69% (from 4119 to 3226), and for the second dataset, the reduction was by 26.93% (from 170 259 to 124 400). Finally, the evaluation results from Table 1 confirm that ThreatTrace can improve the performance of ML

Table 1. Results of two prediction models for various data representations and the baselines reported in [8] and [13] for FedCSIS 2023 Challenge and IEEE BigData 2019 Cup data, respectively. In addition to the ROC AUC values, the column *embedding size* indicates the sizes of activity embeddings used in ThreatTrace.

FedCSIS 2023 Challenge data			
representation type	embedding size	GLMnet	XGBoost
baseline from [8]	–	–	0.6910
basic feat. extraction	–	0.5400	0.8913
ThreatTrace	4	0.7123	0.9067
ThreatTrace	16	0.7393	0.9245
ThreatTrace	32	0.7459	0.9395
IEEE BigData 2019 Cup data			
representation type	embedding size	GLMnet	XGBoost
baseline from [13]	–	–	0.8704
basic feat. extraction	–	0.8351	0.9014
ThreatTrace	8	0.8675	0.9194
ThreatTrace	24	0.8791	0.9284
ThreatTrace	40	0.8807	0.9273

models in detecting cyber threats. The statistical significance of these results was checked and reaffirmed using the DeLong test [9] for comparing ROC curves.

5 Limitations and Future Works

Although we have demonstrated that ThreatTrace enables learning useful representations of complex processes and facilitates training of accurate anomaly detection models, it should not be regarded as a one-size-fits-all solution. Our approach to modelling traces through compacting activity sequences and associating them with Gaussian-distributed clusters is motivated by the continuous (i.e., without a well-defined beginning and end, with many possible activity types) and inherent stochastic nature of the underlying processes. As such, ThreatTrace may not be suitable for the analysis of typical business process models.

Moreover, ThreatTrace captures the *directly follows* relation between activities through the identification of frequent sequential patterns and the use of a trace compacting technique. This approach, however, doesn't allow us to identify global dependencies between activity sequences, such as the *eventually follows* relation. This is caused by the fact that when constructing representations of individual traces from activity embeddings, we neglect their ordering. It is worth noting that in ThreatTrace, global dependencies, such as the *eventually follows* relation, are partially reflected in the activity embeddings. The GloVe algorithm [18] uses a fixed context window to count co-occurrences of activities. Thus, the proximity of activities in a trace is captured in their representation. However, we acknowledge that this problem requires a more thorough investigation in future research. One possible way to address this issue is to modify

the embedding learning algorithm to consider the global position of activities in traces. In NLP, it is commonly done by using positional encoding techniques [24], and we believe that a similar approach can be adapted by ThreatTrace.

We also consciously simplify the trace representation by only considering the diagonal of the covariance matrix during the aggregation of activity embeddings. In practice, such simplification is equivalent to assuming the independence of embedding dimensions, which is usually not true. It limits the shapes of activity embedding clusters that we can accurately summarise with Gaussian distributions. To consider whole covariance matrices, we would have to keep track of sums of all products of embedding dimension pairs. As a result, maintaining the trace representation would require $O(k^2)$ computations and memory, and it would be impractical for larger k values. We noticed, however, that increasing the dimensionality of embeddings after a certain point brings diminishing returns. In future, it would be worthwhile to explore the possibility of using smaller activity embeddings and full covariance matrices.

Finally, we acknowledge the need for further validation of ThreatTrace with diverse datasets to increase its robustness and applicability across various environments. In particular, we see the need to further explore algorithms for the automatic extraction of sequential patterns from data and optimise the sorting of rules used for trace compacting. Combined with previously mentioned ideas, this research direction holds the potential to further refine our methodology and contribute to the ongoing efforts to advance cybersecurity practices.

6 Concluding Remarks

Our research introduces a promising direction for enhancing cybersecurity analysis in network and IoT ecosystems through the application of process mining. By analysing sequences of activities performed by system programs and network events, we demonstrated the benefits of using ThreatTrace for detecting cyber-attacks on IoT devices and recognising truly suspicious network activities. Our method combines frequent pattern-based trace compacting with a novel way of representing traces of continuous processes as multivariate Gaussian distributions defined in the activity embedding space. From a theoretical perspective, we showed that clustering such distributions using the Wasserstein distance is equivalent to clustering our representations in the Euclidean space. We used this fact to construct soft clusters of process traces and utilise the fuzzy cluster membership information as features for predictive models. Finally, through our experiments, we demonstrated that ThreatTrace helps to improve the quality of ML models for detecting cybersecurity threats.

In summary, our study underscores the significance of process mining in fortifying security measures for networks and IoT devices. By leveraging the process view of cyber-systems, we pave the way for proactive detection and mitigation of cyber threats, ultimately fostering safer and more resilient systems.

References

1. van der Aalst, W.M.P.: Process Mining – Data Science in Action, Second Edition. Springer (2016)
2. Adamczyk, B., et al.: Dataset generation framework for evaluation of IoT Linux host–based intrusion detection systems. In: 2022 IEEE Int. Conf. on Big Data (Big Data). pp. 6179–6187 (2022)
3. Batagelj, V.: Generalized ward and related clustering problems. Classification and related methods of data analysis **30**, 67–74 (1988)
4. Bodyanskiy, Y., Boiko, O.: Online Fuzzy Clustering of Data Streams, pp. 211–241. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-35480-0_5, https://doi.org/10.1007/978-3-030-35480-0_5
5. Bose, R.P.J.C., van der Aalst, W.M.P.: Context Aware Trace Clustering: Towards Improving Process Mining Results. In: SDM 2009. SIAM (2009)
6. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 785–794. KDD '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939785, https://doi.org/10.1145/2939672.2939785
7. Clement, P., Desch, W.: An elementary proof of the triangle inequality for the wasserstein metric. Proceedings of the American Mathematical Society **136**(1), 333–339 (2008). https://doi.org/10.1090/S0002-9939-07-09020-X, https://www.ams.org/journals/proc/2008-136-01/S0002-9939-07-09020-X
8. Czerwiński, M., et al.: Cybersecurity Threat Detection in the Behavior of IoT Devices: Analysis of Data Mining Competition Results. In: FedCSIS 2023. ACSIS, vol. 35 (2023)
9. DeLong, E.R., DeLong, D.M., Clarke-Pearson, D.L.: Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. Biometrics **44**(3), 837–845 (Sep 1988)
10. Evermann, J., Rehse, J.R., Fettke, P.: Predicting process behavior using deep learning. Decision Support Systems **100**, 129–140 (2017)
11. Gómez, J.R., et al.: An automatic complex event processing rules generation system for the recognition of real-time IoT attack patterns. Eng. Appl. Artif. Intell. **123**(Part B), 106344 (2023). https://doi.org/10.1016/J.ENGAPPAI.2023.106344
12. Hashemi, S.E., Gholian-Jouybari, F., Hajiaghaei-Keshteli, M.: A fuzzy c-means algorithm for optimizing data clustering. Expert Syst. Appl. **227**(C) (Oct 2023). https://doi.org/10.1016/j.eswa.2023.120377, https://doi.org/10.1016/j.eswa.2023.120377
13. Janusz, A., Kałuża, D., Chądzyńska-Krasowska, A., Konarski, B., Holland, J., Ślęzak, D.: IEEE BigData 2019 Cup: Suspicious Network Event Recognition. In: 2019 IEEE International Conference on Big Data (IEEE BigData), Los Angeles, CA, USA, December 9-12, 2019. pp. 5881–5887. IEEE (2019). https://doi.org/10.1109/BIGDATA47090.2019.9005668, https://doi.org/10.1109/BigData47090.2019.9005668
14. Koninck, P.D., et al.: act2vec, trace2vec, log2vec, and model2vec: Representation Learning for Business Processes. In: BPM 2018. LNCS, vol. 11080, pp. 305–321. Springer (2018)
15. Macák, M., Daubner, L., Sani, M.F., Buhnova, B.: Cybersecurity analysis via process mining: A systematic literature review. In: ADMA. LNCS, vol. 13087, pp. 393–407. Springer (2021)

16. McInnes, L., Healy, J., Melville, J.: Umap: Uniform manifold approximation and projection for dimension reduction (2020), https://arxiv.org/abs/1802.03426
17. Mikolov, T., et al.: Distributed representations of words and phrases and their compositionality. In: NIPS'13 – Vol. 2. Curran Associates Inc., USA (2013)
18. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: EMNLP. pp. 1532–1543 (2014)
19. Song, M., et al.: Trace clustering in process mining. In: BPM Workshops. Revised Papers. LNBIP, vol. 17, pp. 109–120. Springer (2008)
20. Tay, J.K., Narasimhan, B., Hastie, T.: Elastic net regularization paths for all generalized linear models. Journal of Statistical Software **106**(1), 1–31 (2023). https://doi.org/10.18637/jss.v106.i01
21. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Predictive business process monitoring with structured and unstructured data. In: International Conference on Business Process Management (BPM). pp. 401–417. Springer (2019)
22. Vázquez-Barreiros, B., et al.: Prodigen: Mining complete, precise and minimal structure process models with a genetic algorithm. Inf. Sci. **294**, 315–333 (2015)
23. Wang, C., Nulty, P., Lillis, D.: A comparative study on word embeddings in deep learning for text classification. In: NLPIR. p. 37–46. Association for Computing Machinery (2021)
24. Wang, Y.A., Chen, Y.N.: What Do Position Embeddings Learn? An Empirical Study of Pre-Trained Language Model Positional Encoding. In: Webber, B., Cohn, T., He, Y., Liu, Y. (eds.) Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 6840–6849. Association for Computational Linguistics (Nov 2020). https://doi.org/10.18653/v1/2020.emnlp-main.555, https://aclanthology.org/2020.emnlp-main.555
25. Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE Transactions on Neural Networks **16**(3), 645–678 (2005)
26. Zaki, M.J.: SPADE: an efficient algorithm for mining frequent sequences. Mach. Learn. **42**(1/2), 31–60 (2001)
27. van Zelst, S., et al.: Event abstraction in process mining: Literature review and taxonomy. Granular Computing **6**, 719–736 (Jul 2021)
28. Zhong, Y., Goulermas, J.Y., Lisitsa, A.: Process mining algorithm for online intrusion detection system. In: Yavorskiy, R., Cavalli, A.R., Kalenkova, A. (eds.) Tools and Methods of Program Analysis. pp. 15–25. Springer Nature Switzerland, Cham (2024)