Cyber-Attack Detection through Trace Abstraction and Hierarchical Clustering

Abstract. The growing availability and application areas of the Internet of Things (IoT) have raised the importance of providing a proper level of cybersecurity to networks of interconnected devices. Process mining gives us an opportunity to analyze sequences of actions performed by system programs on such devices to detect cyber-attacks from a novel perspective, i.e., as continuous processes that are composed of consecutive system calls. However, datasets representing these continuous processes are usually characterized by large variability and complexity. We propose a method for simplifying complex process traces that can often be observed in audit logs of system operations in domains such as cybersecurity. Our method combines the ideas of trace abstraction and clustering to generate compact process variants that preserve the behavior of interest - i.e., patterns of potential cyber-attacks. The first idea is based on simple compacting rules that can either be manually designed by experts or automatically extracted from data using popular frequent sequence mining algorithms. The second idea uses Ward's linkage agglomerative clustering to identify a relatively small number of process variant clusters and their most influential representatives. We then train machine learning models to detect cyber-attacks in these compacted audit logs. In addition to experimental results demonstrating the usefulness of such representatives for the early detection of cyber-attacks on IoT devices, we justify the selection of Ward's method for clustering in combination with the proposed process variant dissimilarity measure. The experimental results show that our approach performs better than current cyber-attack detection approaches that do not take the process perspective into account.

Keywords: Process Mining · Cybersecurity · Cyber-Attack Analysis · Trace Abstraction · IoT Data.

1 Introduction

The security landscape of Internet of Things (IoT) devices is increasingly critical due to our reliance on connected technology. Process mining [1] offers a fresh perspective by analyzing sequences of actions performed by 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, IoT devices exhibit regular behavior, making deviations from the norm potential indicators of cyber-attacks. In practice, however, system processes are challenging to analyze due to their inherent complexity and variability. Process mining

methods can empower cybersecurity analysts to trace attack paths and 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 behavior. 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 for proactive cybersecurity measures.

In this research, we aim to address the above-mentioned challenges. We propose a method for the analysis of audit logs from short time windows of system operations that allows accurate detection of cyber-attacks. Our approach combines a rule-based trace abstraction technique with hierarchical clustering and heuristics for choosing the most characteristic cluster representatives. We use the process variants resulting from our analysis to construct vector representations of system log sequences. We then train machine learning models to detect cyber-attacks in newly observed logs. The experimental results show that our approach performs better than current cyber-attack detection approaches.

Our paper is structured as follows. Section 2 discusses related work. Then, in Section 3, we explain our novel approach to process variant abstraction, i.e., in Subsection 3.1, we describe our rule-based approach to trace compacting, in Subsection 3.2 we provide a rationale behind the use of an agglomerative clustering algorithm combined with Ward's linkage function. Then in Subsection 3.3, we explain the variant dissimilarity function that is used for clustering and data visualization, and in Subsection 3.4, we recount the heuristics for selecting cluster representatives. In Section 4, we describe the task and data that we use in this study, and then, in Section 5 we share the results of our experiments in which we investigated the impact of the proposed approach on the quality of two popular prediction algorithms for detecting cyber-attacks on IoT devices. Finally, in Section 6, we conclude the paper and point to future work.

2 Related Work

In this section, we discuss process mining applications in cybersecurity and existing techniques for trace abstraction and hierarchical clustering of traces.

Process mining techniques (particularly process discovery and conformance checking methods) have been applied to various areas in cybersecurity [22], including industrial control systems, where it is used for the detection of cyber attacks [25]. In network security, process mining methods have been combined with hierarchical clustering [3] and real-time analysis [9] to improve intrusion alert visualization, prioritization of novel IoT attacks, and understanding of net-

work behaviors and DNS trace anomalies [8]. Additionally, web application security uses genetic process mining [28] and workflow model discovery [10] to identify abnormal user activities, enforce security policies, and detect deviations during audits [35]. Process discovery and conformance-checking techniques have proven to be beneficial in the field of attack inspection as a means of gaining insight into attack strategies [30] and outlier user behavior to identify anomalous user behavior and malicious insiders [27].

The fine-grained and complex nature of real-world process traces often culminates in "spaghetti-like" process models [1]. To mitigate this, event abstraction and reduction methodologies have been employed, streamlining process models to enhance interpretability [34]. Semi-automatic event-activity matching [4] and guided process discovery [23] show how low-level events can be aligned with high-level activities. Unsupervised methods such as clustering [19] and the use of a pattern taxonomy [6] facilitates the automation of the abstraction process, helping to identify structured patterns and high-level activities within the data. Additionally, techniques involving partially ordered event data [20] and behavioral process mining [14] also help reveal abstract pattern instances and subprocesses to improve precision in process models. These evaluated techniques [16] highlight the necessity of choosing the right abstraction methods, balancing fitness, precision, and simplicity.

Agglomerative hierarchical clustering is a bottom-up approach that starts with each data point as a separate cluster and merges the most similar clusters until a single cluster remains [26]. Existing studies explore the application of agglomerative hierarchical clustering in process mining, along with other techniques (e.g., sequence alignment, contextual data incorporation, etc.), to further cluster similar traces together based on distance metrics [7, 29, 32]. These methods deal with process variability and noise in event logs. Among various linkage methods (i.e., determinants of the similarity between clusters), there is research evidence for the use of Ward's linkage method in process mining as it minimizes the total within-cluster variance, resulting in compact and well-separated clusters [15, 18].

This paper proposes a novel combination of abstraction and hierarchical clustering techniques to simplify complex process models for more effective security analysis. This sets our work apart from most reviewed studies, which do not delve deeply into abstraction and clustering methods to aid in detecting cybersecurity attacks. However, our approach does share similarities with some of the reviewed work, e.g. Rodríguez et al. [27] which uses process discovery to identify anomalous behaviors, Myers et al. [25], which leverages conformance checking to detect deviations from normal processes, and de Alvarenga et al. [3], which employs hierarchical clustering to acquire meaning from complex traces. Motivated by the use of process mining to extract valuable insights from intricate log data to strengthen cybersecurity defenses, this study contributes to the growing body of knowledge on process mining and cybersecurity by incorporating an unsupervised abstraction strategy and hierarchical clustering with the Ward linkage criterion for process trace reduction.

3 Constructing Abstractions of Process Traces

When dealing with real-world processes in complex domains such as cybersecurity, standard approaches to process modeling may face the problem of large variability in the observed process traces. As a result, constructed process models such as process graphs can become overly large and difficult to analyze by domain experts. One possible way to overcome this issue is to simplify the observed process traces by considering frequently repetitive sequences of consecutive actions as individual blocks that can be used as new nodes in the process graph. Another possibility is to cluster observed process traces into groups of highly similar variants and focus the analysis only on the most representative examples from each group. In this paper, we propose combining these two approaches to discover a compact set of trace variants that facilitates the analysis of the underlying process.

3.1 Compacting of process variants

Let us denote the j-th process trace in available data by a sequence of action symbols $T^j = [X_1, X_2, \ldots, X_{N_j}]$, where each X_i is a symbol corresponding to one of the possible actions from the set A, i.e., $X_i \in A = \{A_1, \ldots, A_n\}$. 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 set 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 the 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 provided by domain experts or could be automatically extracted from data using an arbitrary frequent sequence mining algorithm such as SPADE [33]. In the latter case, it is typically beneficial to sort the list R decreasingly by rule support. In this way, rules that are closer to the beginning of the list likely contribute more to the reduction of process traces. However, more advanced optimization heuristics, such as the genetic algorithm [31], can also be used to decide the ordering of the compacting rule set.

Our overall goal for this step is to reduce the number of variants that are essentially the same from a cyber-attack point of view while preserving outliers that may be indicators of potential attacks.

3.2 Hierarchical clustering of process variants

After obtaining compacted process trace variants, we may further simplify our model of the observed process by clustering these variants and selecting the most representative examples for further analysis. The number of compacted variants is typically much lower than the total number of process traces in the original data. Thus, to cluster the compacted variants we can even use computationally complex algorithms, such as the agglomerative nesting (agnes) method [26].

Agglomerative nesting is a hierarchical clustering algorithm in which the hierarchy of clusters is constructed in a bottom-up fashion. At the beginning, each data point is a distinct cluster. Then, at each consecutive step, the two closest clusters are merged until all data belong to a single cluster. The proximity of clusters is determined using a function often called the linkage function, e.g., single-link, average-link, or complete-link (the minimum, average, or maximum distance between pairs of points from two clusters, respectively). After trying the aforementioned linkage functions, we found that Ward's linkage works best for the clustering of trace variants. This function is defined as:

$$\begin{aligned} D^W(C_i, C_j) &= \frac{|C_i| \cdot |C_j|}{|C_i| + |C_j|} ||\mu_{C_i} - \mu_{C_j}||^2 \\ &= \sum_{x \in C_i \cup C_j} ||x - \mu_{C_i \cup C_j}||^2 - \sum_{x \in C_i} ||x - \mu_{C_i}||^2 - \sum_{x \in C_j} ||x - \mu_{C_j}||^2 \end{aligned} \tag{1}$$

where μ_{C_i}, μ_{C_j}, $\mu_{C_i \cup C_j}$ are the centroids of clusters C_i, C_j, $C_i \cup C_j$, respectively.

Although the original definition of Ward's linkage uses squared Euclidean distance, it has been shown in [5] that it can be easily generalized to any dissimilarity metric. Let us denote by $\hat{\mu}_C$ a generalized (possibly imaginary) centroid of cluster C. If we consider any dissimilarity d, then the generalization of equation (1) takes the form:

$$\begin{aligned} D^W(C_i, C_j) &= \frac{w(C_i) \cdot w(C_j)}{w(C_i \cup C_j)} d(\hat{\mu}_{C_i}, \hat{\mu}_{C_j}) \\ &= \frac{1}{w(C_i \cup C_j)} \sum_{x \in C_i, y \in C_j} w(x) \cdot w(y) \cdot d(x, y) - \frac{p(C_i)}{w(C_i)} - \frac{p(C_j)}{w(C_j)} \end{aligned} \tag{2}$$

where $w(x)$ is a weight of a data point x, which in our case equals the number of process traces corresponding to the given variant, and $w(C)$ is the weight of cluster C, which for us is a sum of weights of all variants $x \in C$. The function p in formula (2) is the generalized version of the Ward criterion function:

$$\begin{aligned} p(C) &= \frac{1}{w(C)} \sum_{x \in C} w(x) \cdot d(x, \hat{\mu}_C) \\ &= \frac{1}{2 \cdot w(C)} \sum_{x, y \in C} w(x) \cdot w(y) \cdot d(x, y). \end{aligned} \tag{3}$$

This generalization of Ward's linkage keeps all properties necessary for the application of the Lance-Williams method [5] which guarantees that at any iteration

of the agglomerative nesting algorithm, we can find a pair of clusters whose linkage function value is minimal (and compute that minimal value) using only the information about pairwise cluster distances from the previous iteration. Thus, the algorithm that uses it for constructing a hierarchy of clusters remains unchanged.

3.3 Computing dissimilarities between process variants

To conduct a hierarchical cluster analysis of process traces using Ward's linkage function it is necessary to compute pairwise dissimilarities between trace variants. We can achieve this by either selecting a metric that is appropriate for comparing various-length sequences of discrete actions (see Subsection 3.1) or by computing embeddings of trace variants in a metric space R^k and using standard Euclidean distance.

The first approach is more direct and we took it in our case study. We define the dissimilarity function d as a normalized Levenshtein distance:

$$d(T^i, T^j) = \frac{lev(T^i, T^j)}{\mid T_i \mid + \mid T_j \mid} \tag{4}$$

where $lev(x, y)$ is the standard Levenshtein distance. One of the advantages of using this measure is that its values are always in the $[0, 1]$ interval. Additionally, it allows computation of the dissimilarity between trace variants of different lengths and preserves the dissimilarity scale when pairs of variants are compared.

It is worth noting that an alternative approach could involve using some well-established techniques from the NLP domain. For example, the *word2vec* model [24] could be used to create embeddings of possible actions from A and A', and then the embeddings could be aggregated over compacted process variants to embed them in R^k. If our data is sufficiently large, we could also train a deep bidirectional transformer model [13] and use its encoder part to compute the R^k embeddings of whole sequences corresponding to process trace variants.

3.4 Selecting variant cluster representatives

While performing a cluster analysis, it is useful to investigate the most representative members of each cluster. To further reduce the analyzed set of process trace variants, we consider a few different criteria for that task. The first one is aimed at selecting the most central data points. From each cluster C, we choose the variants with the lowest value of the marginality function:

$$marginality(T) = \sum_{t \in C_T} w(t) \cdot d(t, T) \tag{5}$$

where C_T is the cluster that contains T and $w(t)$ is a weight associated with t.

The second criterion focuses on the commonness of individual trace variants. From each cluster, we simply choose the variants with the highest weight relative

to the weight of the cluster:

$$commonness(T) = \frac{w(t)}{w(C_T)} \tag{6}$$

Additionally, when each process trace instance can be associated with a label from a finite set $L = \{0, \ldots, l\}$, we may consider choosing cluster representatives that correspond to variants with the highest conditional probability of particular labels. For example, in our case study, we consider a label indicating whether a cyber-attack was ongoing during the time when we observed a particular process instance. Since different process trace instances that correspond to the same trace variant could have a positive or negative label value, we may estimate the probabilities of each label for all variants observed in the historical data. Then, we may use those estimations to choose the most characteristic variants for each label in every cluster.

4 Case Scenario: Cyber-Attacks on IoT Devices

Our case study in this work is related to a vital cybersecurity problem, i.e., the detection of malicious activity in the operations of IoT devices. The data set that we use comes from a recent international data science competition organized at KnowledgePit.ai web platform[1]. In our work, we slightly modify the task from [11] to avoid data leak issues that haunted the original competition setup. We explain the task and highlight the differences that we introduced.

4.1 Problem description

The original data set comprised data 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 [11]. It was created using an experimental environment developed as a result of a project conducted by three companies, i.e., EFIGO, EMAG, and QED Software [2], focused on the cybersecurity of IoT devices. The entire environment was isolated on a separate network to ensure that no external factors interfered with the experiment. To obtain a sample that contained both regular and anomalous device operations, it was necessary to model the normal operation of the device and simulate cyber-attacks. 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 external cyber-attacks on the device.

Typical operating conditions were generated continuously in several independent ways. This involved SSH sessions, where an administrator logged into the device and ran several system commands with intervals ranging from 0.5 to 11.5 seconds. HTTP WAN traffic was also simulated using the device's built-in HTTP server to send cyclic queries. The queries were based on real WAN-connected

[1] https://knowledgepit.ai/fedcsis-2023-challenge/

devices, and their intervals were taken from historical data. Additionally, a file transfer service was run on the device to simulate a periodic software update. A binary file of varying sizes (from 512 to 1,024 bytes) was sent with a random interval of 1 to 12 hours. The device also had a dedicated endpoint for outer status checking and device clock synchronization, allowing specialized HTTP queries to be sent. These queries ran the "date -date now" command and were released with randomized intervals.

Two different types of attacks were simulated, i.e., the remote code execution and path traversal. In the first attack, a vulnerable endpoint "clock.php" was exploited, and a command injection vulnerability was used to invoke a query. This attack was aimed at establishing a reverse connection with the attacking host and running an interactive session of the console "sh" program. Random commands were then invoked at irregular intervals. In the second attack, a path traversal vulnerability was used to upload a file into an unusual location on the device's local file system. The file names were random, as were their sizes (from 20 to 5,024 bytes). The number of files varied between 1 and 10, and the time between uploads ranged from 0.5 to 10 seconds. More details about this experimental environment and the data generation process can be found in [11].

The system audit logs collected from the monitored IoT device were divided into 1-minute-long time windows and automatically labeled using information about the timings of scheduled attacks. Different types of attacks were not distinguished – only binary information about the presence of an attack was kept. As a result, the acquired data can be used for benchmarking the performance of algorithms for detecting cyber-attacks on IoT devices. Since the data set is composed of sequences of raw system audit logs, it allows for an objective comparison of various approaches to data modeling. In particular, classical machine learning techniques can be applied in combination with data aggregation and feature extraction methods [21]. Alternatively, recurrent neural networks or temporal transformers can be applied to model the sequences of system logs. In our study, however, we decided to utilize the process mining approach to facilitate the explainability and reliability of the resulting cyber-attack detection model.

4.2 Data characteristics

The raw data was collected from the monitored IoT device in the form of system audit logs. These logs were parsed and transformed into CSV tables corresponding to individual 60-second time windows. We will refer to each such period as a single data case. Thus the data set was composed of two sets of files, i.e., training data composed of 15027 CSV tables and test data containing 5017 tables. The original division in the data science competition was based on time, i.e., test tables corresponded to time periods strictly after the training data, and we kept this division in all our experiments. Additionally, we further divide the training data into two disjoint sets – one with 5009 data cases that we use for discovering process variants and the second set, with 10018 data cases, that we use for training ML models for detecting cyber-attacks.

Each CSV table was labeled as *normal* or *cyber-attack* depending on whether a cyber-attack was conducted during the corresponding 60 seconds. The rows of each table corresponded to individual system events recorded in the corresponding period. Events were described by 40 attributes that included 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 some higher-level features extracted from the system audit logs such as counts of system kernel calls or the number of active services. We show an exemplary snippet of one of the available CSV tables in Appendix A included in supplementary materials[2].

In our study, we analyze this data from the perspective of process science. We focus on system calls associated with particular PIDs, i.e., we associate each sequence of system calls (actions) with the same PID in the 60-second period with a single instance (a case) of the investigated process. In this way, we have instances of multiple observed processes in each of the available CSV tables. The average number of process instances in a single table is 32.36 and the average number of events associated with a single process instance is 59.21. The total number of process instances in the training and test data is 648, 539. There are 77 different basic action types in the data and we extend this set to 156 action types through our process variant compacting rules.

It is worth noting that the original data set used in the competition at KnowledgePit.ai 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 for every execution were given the same PIDs. 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 independently randomize PID assignments in every considered time window. In this way, we avoid an undesired information leak and make the conducted experiments more realistic.

5 Experimental Results

We demonstrate how the rule-based compacting of process trace variants combined with Ward's linkage-based clustering method improves the detection of cyber-attacks on IoT devices. We analyze the data from FedCSIS 2023 Challenge with a focus on the process mining aspect and we evaluate its impact on the performance of ML models for the considered task. Preprocessed data used in our experiment, as well as the source code and results, are available as a part of the supplementary materials in our open repository[2].

[2] The supplementary materials are available at https://bit.ly/Cyber-Attack-Detection

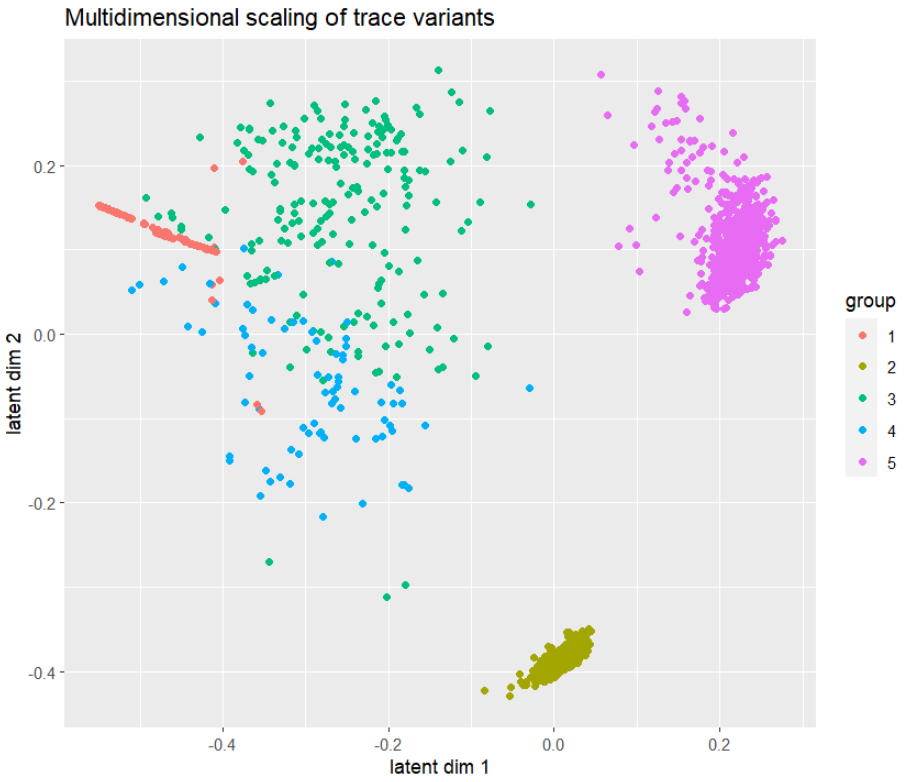

Fig. 1. Visualization of the discovered process variant clusters.

5.1 Discovering the most representative process trace variants

In the first part of our experiment, we investigated the diversity among process traces in the available data. As we described in Subsection 4.2, we used the process discovery part of training data (159, 721 process traces) to identify the most representative process variants. It was done using our own implementation inspired by the *bupaR* library for R language [17]. In total, we discovered 3, 351 distinct process variants that considerably differed in length and support size (i.e., the number of associated process traces). The length varied between 1 and 60, 920 with a mean value of 197.8 and a median of 145. The supports ranged between 1 and 27, 387 with a mean value of 47.66 and a median of 1 – in fact, over 75% traces were observed only once in our data. Such large variability makes it difficult to thoroughly analyze the data and create meaningful process models. To exemplify this issue, we include a figure showing a spaghetti-like process graph visualization constructed for the original data as supplementary materials in our repository[2].

To tackle the observed trace variability, we applied the manually constructed trace compacting rules. Our rule set was composed of a rule that assigns a new

identifier to pairs of system calls that open and close a connection to a file, and three rules for each action a: $([aa] \rightarrow a')$, $([a'a] \rightarrow a')$, $([a'a'] \rightarrow a')$, that were responsible for reducing any sequence of two or more actions of the same type to a single new action ID (e.g., a sequence [aaaaa] is compacted into [a']). Thus, in total, our rule set contained $3 \times (77 + 1) = 234$ simple rules.

As a result, we obtained a reduced set of $1,450$ process variants whose average support was 110.2. However, the variant set was still diverse, and over 60% of variants corresponded only to a single trace. Since our task is to detect relatively rare cyber-attacks, we could not discard such unique variants as they might be indicative of unusual system behavior. Thus, in the second step, we clustered the variants using the method proposed in Subsection 3.2. We started by computing a dissimilarity matrix D using the formula (4). This matrix was used not only for the clustering but also to compute the MDS embeddings [12] for data visualization purposes. Figure 1 shows the resulting division of data into five variant clusters in a two-dimensional embedding space.

The last step in our analysis was the selection of the most characteristic process variants. We sampled each of the discovered clusters using the four criteria described in Subsection 3.4, namely the *marginality*, *commonness*, as well as high, and low conditional probabilities of a cyber-attack. Figure 2 shows a visualization of the cyber-attack probability distribution for variants in the process discovery part of our data. The supports of individual variants are marked by the size of corresponding dots and the cyber-attack probabilities are expressed by the heatmap. Overall, after removing duplicates resulting from choosing the same variants based on different criteria, we selected 78 variants to represent processes related to operations of the monitored IoT device. The average support of these variants was 1607.705 and only 11% of them were supported by a single trace. Figure 3 shows a comparison between centile function values for the tree variant abstraction levels, namely the initial process variants, compacted variants, and the variants selected as cluster representatives.

5.2 Detecting cyber-attacks

We conducted a number of experiments to verify how various representations influence the performance of prediction models for detecting cyber-attacks. We focused on representations stemming from our analysis of system processes and measured the impact of the proposed trace abstraction strategy.

Since the labeled cases in our data correspond to 1-minute periods in which we observe actions of all system processes, the most straightforward representation that we examine is a binary vector of a length equal to the number of considered process variants. Each element of such a vector indicates whether a particular variant was observed in the corresponding period. To evaluate the influence of the proposed trace abstraction method, we considered four different sets of variants for constructing representations: 1) all variants without any trace compacting, 2) all compacted variants observed in the model-training part of our data, 3) compacted variants observed in both the process discovery and

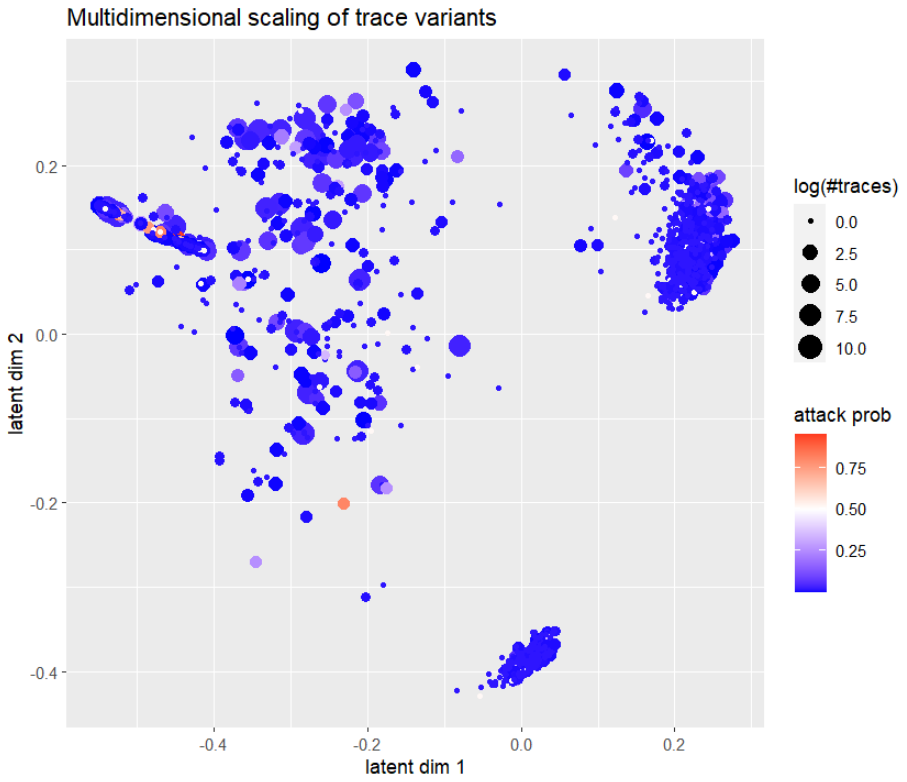

Fig. 2. Visualization of discovered process variants with a cyber-attack conditional probability heatmap.

model-training parts of data, and 4) compacted variants selected as the most characteristic representatives of the discovered variant clusters.

We combined the binary representation with basic statistics computed for the variants observed in each period. The set of considered statistics included, e.g., the number of observed traces, and the maximum, mean and minimum size of the observed traces before and after compacting. It is worth mentioning that the utilized set of additional trace characteristics could be easily extended to potentially further improve the quality of the resulting models. Additionally, for the fourth representation based on cluster representatives, we associated each data case with a vector of minimal dissimilarities between the selected variants and the variants observed in the corresponding time period. The final dimensionality of the resulting representations is given in Table 1.

We use the described representations to fit two types of prediction models for detecting cyber-attacks. The first one is the decision tree gradient boosting model implemented in the *XGBoost* library[3]. The second model is the LASSO-

[3] https://xgboost.readthedocs.io/en/stable/

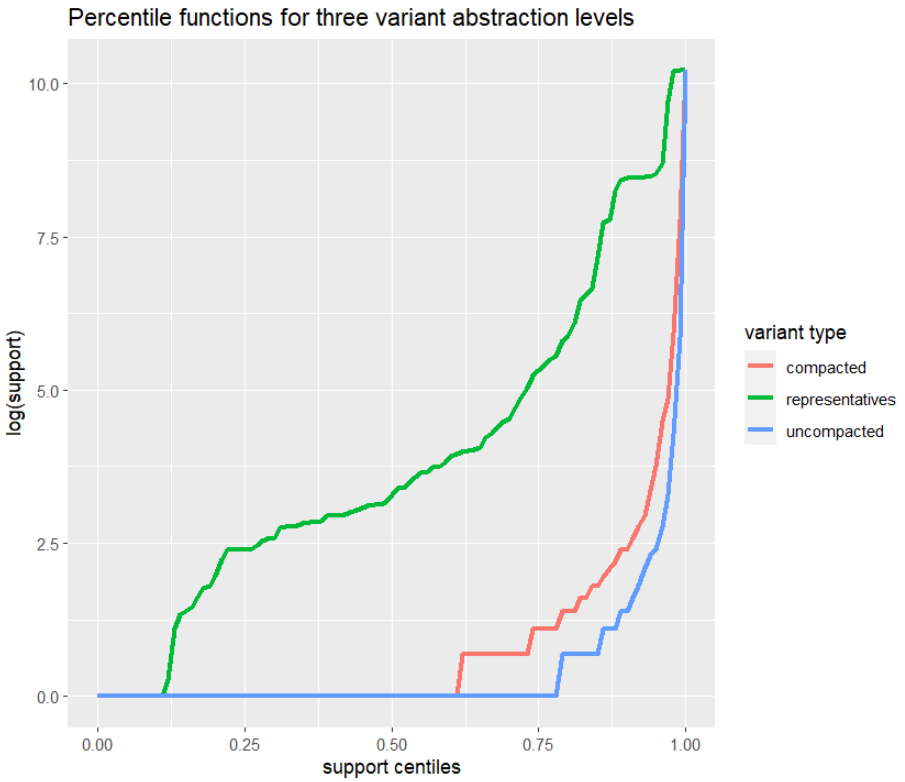

Fig. 3. Visualization of support percentile function values for the initial process variants, compacted variants, and the variants selected as cluster representatives.

Table 1. Results of two cyber-attack detection models for the four considered data representations and the baseline reported in [11]. In addition to the achieved ROC AUC values, the column *rep. size* indicates the sizes of the corresponding representations.

representation	rep. size	XGBoost	GLMnet
baseline model from [11]	–	0.6910	–
all variants	5769	0.9096	0.9061
compacted variants observed from model training	2114	0.9238	0.6311
compacted variants observed from discovery & training	1468	0.9259	0.7062
compacted variants that are cluster representatives	96	0.9325	0.7710

regularized logistic regression implemented in the *glmnet* library[4]. The reason for using these two classifiers is that while gradient boosting can be regarded as state-of-the-art in the tabular data classification task, logistic regression is still a common choice in practice due to its high power efficiency. The simplicity

[4] https://glmnet.stanford.edu/

and low computational cost of inference make the logistic regression suitable for operating on mobile IoT devices.

One setting of hyperparameters for each of the models was used for all considered representations. The gradient boosting model used 1000 trees whose maximal depth was set to 8. The regularization parameters *alpha* and *lambda* were set to 10.0 and 1.0, respectively. Additionally, column sampling was used with a sampling rate of 0.75 to avoid overfitting. Other settings were not changed from their default values. For the GLMnet model, we set the *alpha* value to 1.0, i.e., only the LASSO regularization was used. The *lambda* value was automatically tuned based on cross-validation results.

Table 1 shows test ROC AUC results obtained for each of the considered representation methods and the original baseline model that was reported in [11] (the version that was not exploiting the issue discussed in Subsection 4.2). Noticeably, all models that used information extracted from the analysis of process variants achieved considerably higher scores than the baseline model. A data representation used by the baseline was based on aggregations of original feature values from the raw system audit logs combined with embeddings of alphanumeric features computed using NLP techniques. Compared to that representation, our approach is not only less complicated but also leads to higher ROC AUC scores. The differences in results between the four considered representations show the impact of the proposed trace abstraction method. The XGBoost model obtained similar results for the representations based on compacted variants from training data and compacted variants from the process discovery data part. These results were $\approx 1\%$ higher than the score obtained by the representation that was not using compacted process variants and $\approx 1\%$ lower than the score achieved by the representation based on representatives of variant clusters. Greater differences were observed for the GLMnet model for which the best results were noted for the representation based on all variants, without any compacting. Nevertheless, even in that case, the score is considerably lower than the result of the proposed representation method and XGBoost model. This difference demonstrates the impact of the proposed variant compacting and clustering approach. It also highlights the usefulness of incorporating the process mining view for the cyber-attack detection task.

6 Conclusions and Future Works

Our research introduces a promising direction for enhancing cybersecurity in IoT ecosystems through the use of process mining. By analyzing sequences of actions performed by system programs, we demonstrated the feasibility of our approach in detecting cyber-attacks on IoT devices. Our method, which combines rule-based process variant abstraction with hierarchical clustering techniques, has shown great potential in simplifying complex process traces commonly observed in the cybersecurity domain. We demonstrated how the process variants selected as the most characteristic representatives of variant clusters can be used

for constructing vector representations of system log sequences and how such representations enable the construction of better cyber-attack detection models.

While our experiments showcase promising results, we acknowledge the need for further validation with diverse datasets to increase the robustness and applicability of our approach across various IoT environments. Additionally, we see the need for exploring algorithms for the automatic extraction of compacting rules from data and optimizing the sorting of rule sets in our future research. These endeavors hold the potential to further refine our methodology and contribute to the ongoing efforts to advance cybersecurity practices in IoT networks.

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

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. de Alvarenga, S.C., et al.: Process mining and hierarchical clustering to help intrusion alert visualization. Comput. Secur. **73**, 474–491 (2018)
4. Baier, T., et al.: Bridging abstraction layers in process mining. Inf. Syst. **46**, 123–139 (2014)
5. Batagelj, V.: Generalized Ward and Related Clustering Problems. In: IFCS, 1987. North-Holland (1988)
6. Bose, R.P.J.C., van der Aalst, W.M.P.: Abstractions in Process Mining: A Taxonomy of Patterns. In: BPM 2009. LNCS, vol. 5701, pp. 159–175. Springer (2009)
7. 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)
8. Bustos-Jiménez, J., et al.: Applying process mining techniques to DNS traces analysis. In: SCCC 2014. IEEE Computer Society (2014)
9. Coltellese, S., et al.: Triage of IoT Attacks Through Process Mining. In: OTM 2019. LNCS, vol. 11877. Springer (2019)
10. Compagna, L., dos Santos, D.R., Ponta, S.E., Ranise, S.: Aegis: Automatic enforcement of security policies in workflow-driven web applications. In: Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy. pp. 321–328 (2017)
11. 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)
12. Demaine, E., et al.: Multidimensional scaling: Approximation and complexity. In: Proc. of the 38th Int. Conf. on Machine Learning. Proceedings of Machine Learning Research, vol. 139. PMLR (18–24 Jul 2021)
13. Devlin, J., et al.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: North American Chapter of the Assoc. for Computational Linguistics (2019)

14. Diamantini, C., et al.: Behavioral process mining for unstructured processes. J. Intell. Inf. Syst. **47**(1), 5–32 (2016)
15. Hompes, B., et al.: Discovering deviating cases and process variants using trace clustering. In: BNAIC 2015 (11 2015)
16. Houdt, G.V., et al.: An empirical evaluation of unsupervised event log abstraction techniques in process mining. Inf. Syst. **121**, 102320 (2024)
17. Janssenswillen, G.: bupaR: Business Process Analysis in R (2024), version 0.5.3
18. 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)
19. de Leoni, M., Dündar, S.: Event-log abstraction using batch session identification and clustering. In: SAC '20. pp. 36–44. ACM (2020)
20. Li, C., et al.: Event abstraction for partial order patterns. In: BPM 2023. LNCS, vol. 14159, pp. 38–54. Springer (2023)
21. Liu, M., et al.: Gradient boosting models for cybersecurity threat detection with aggregated time series features. In: FedCSIS 2023. ACSIS, vol. 35 (2023)
22. Macák, M., et al.: Process mining usage in cybersecurity and software reliability analysis: A systematic literature review. Array **13**, 100120 (2022)
23. Mannhardt, F., et al.: Guided process discovery - A pattern-based approach. Inf. Syst. **76**, 1–18 (2018)
24. Mikolov, T., et al.: Distributed representations of words and phrases and their compositionality. In: NIPS'13 – Vol. 2. Curran Associates Inc., USA (2013)
25. Myers, D., et al.: Anomaly detection for industrial control systems using process mining. Comput. Secur. **78**, 103–125 (2018)
26. Ran, X., et al.: Comprehensive survey on hierarchical clustering algorithms and the recent developments. Artif. Intell. Rev. **56**(8), 8219–8264 (2023)
27. Rodríguez, M., et al.: Discovering attacker profiles using process mining and the MITRE att&ck taxonomy. In: LADC 2023, 2023. pp. 146–155. ACM (2023)
28. Sahlabadi, M., et al.: Detecting abnormal behavior in social network websites by using a process mining technique. Journal of Computer Science **10**(3), 393 (2014)
29. Song, M., et al.: Trace clustering in process mining. In: BPM Workshops. Revised Papers. LNBIP, vol. 17, pp. 109–120. Springer (2008)
30. Viticchié, A., Regano, L., Basile, C., Torchiano, M., Ceccato, M., Tonella, P.: Empirical assessment of the effort needed to attack programs protected with client/server code splitting. Empir. Softw. Eng. **25**(1), 1–48 (2020)
31. Vázquez-Barreiros, B., et al.: Prodigen: Mining complete, precise and minimal structure process models with a genetic algorithm. Inf. Sci. **294**, 315–333 (2015)
32. Weerdt, J.D., et al.: Active trace clustering for improved process discovery. IEEE Trans. Knowl. Data Eng. **25**(12), 2708–2720 (2013)
33. Zaki, M.J.: Spade: An efficient algorithm for mining frequent sequences. Machine Learning **42**(1–2), 31–60 (Jan 2001)
34. van Zelst, S., et al.: Event abstraction in process mining: Literature review and taxonomy. Granular Computing **6**, 719–736 (Jul 2021)
35. Zerbino, P., Aloini, D., Dulmin, R., Mininno, V.: Process-mining-enabled audit of information systems: Methodology and an application. Expert Systems with Applications **110**, 80–92 (2018)