Comparative Assessment of Process Mining for Supporting IoT Predictive Security

Adrien Hemmer, Mohamed Abderrahim, Rémi Badonnel, Jérôme François and Isabelle Chrisment
Inria Nancy Grand Est - Loria, University of Lorraine
Campus Scientifique, 54600 Villers-les-Nancy, France
{adrien.hemmer, mohamed.abderrahim, remi.badonnel, jerome.francois, isabelle.chrisment}@inria.fr

***Abstract*—The growth of the Internet-of-Things (IoT) has been characterized by the large-scale deployment of sensors and connected objects. These ones are integrated with other Internet resources in order to elaborate more complex systems and applications. Security management is a major challenge for these systems due to their complexity, their heterogeneity and the limited resources of their devices. In this paper we evaluate the exploitability and performance of a process mining approach for detecting misbehaviors in such systems. We describe the considered architecture and detail its operation, from the generation of behavioral models to the detection of potential attacks. We formalize several alternative commonly-used detection methods, including elliptic envelope, support-vector machine, local outlier factor, and isolation forest techniques. After presenting a proof-of-concept prototype, we quantify comparatively the benefits and limits of our process mining solution combined with data pre-processing, through extensive experiments based on different industrial datasets.**

***Index Terms*—Security Management, Internet-of-Things, Process Mining, Data Mining, Machine Learning.**

I. Introduction

THE Internet-of-Things (IoT) has grown in importance and maturity in a large variety of domains, such as domestic applications with smart home networks [1], and industrial infrastructures with the development of industry 4.0 [2]. The complexity of systems and infrastructures involving IoT devices is often under-estimated [3], and induces new challenges from a security management perspective [4]. The weaknesses of IoT-based systems have an impact that goes beyond the Internet-of-Things, and influence other systems that are not composed of such devices. In particular, even if a system does not implement IoT devices, it can be vulnerable to attacks based on infected ones. A typical example can be given with the case of botnets built from compromised IoT devices and serving as a support for distributed denial-of-service attacks (DDoS) [5]. Typically, the Mirai botnet responsible for the series of DDoS attacks against the DynDNS service was composed of such vulnerable IoT devices and caused the unavailability of several major Internet platforms and services during several hours [6]. More recently in 2019, massive botnet attacks exploited more than 400,000 connected devices against online streaming applications [7].

The major risks in IoT-based systems come from the devices themselves that may be affected by naïve weaknesses due to their poor and limited implementations [4][6]. The exploitation of one single weak IoT device can be used to take control over a whole network [8]. Traditional security mechanisms, such as intrusion detection systems (IDS), firewalls and antiviruses, are often inadequate with the constrained resources (CPU, memory, battery) of IoT devices [9]. In addition, these solutions are often specific to given categories of devices and protocols [10], and may fail to address security attacks that occur in complex and heterogeneous environments.

In this paper, we evaluate the exploitability and performance of a process mining approach for detecting misbehaviors in IoT networks. This is an extended version of our work published in [11] where we overview this approach and its architecture, with the generation of behavioral models and the detection of potential attacks. We complement these efforts, with the formalization of four other categories of commonly-used detection methods, including elliptic envelope, support-vector machine, local outlier factor, and isolation forest techniques. We show their integration into the considered predictive security architecture. We quantify the benefits and limits of our process mining method in comparison to these alternative techniques, considering three different industrial datasets in the area of connected cars, industry 4.0 and robot networks. In particular, we show the advantages of combining process mining with data pre-processing, in order to prevent an explosion of states, while keeping high detection performance. The solution is implemented based on a proof-of-concept prototype using the ProM library [12]. It exploits application data generated by IoT devices and is compatible with heterogeneous platforms and protocols. The collection of data is performed passively, and does not introduce additional network and processing overloads at the device level. The objective is clearly to minimize false positive alerts and provide further contextual background to security analysts, while taking advantage of system heterogeneity [13].

The main contributions of this paper include: (1) the design of a predictive security approach for detecting attacks and misbehaviors in IoT-based systems by combining process mining and data pre-processing, (2) the formalization and integration of other commonly-used detection techniques, including elliptic envelope, support-vector machine, local outlier factor, and isolation forest techniques, (3) the development of an operational proof-of-concept prototype exploited for three different industrial datasets such as connected cars, industry 4.0, robot networks and (4) a multi-criteria comparative analysis of the solution performance based on extensive experiments.

The remainder of the paper is organized as follows.

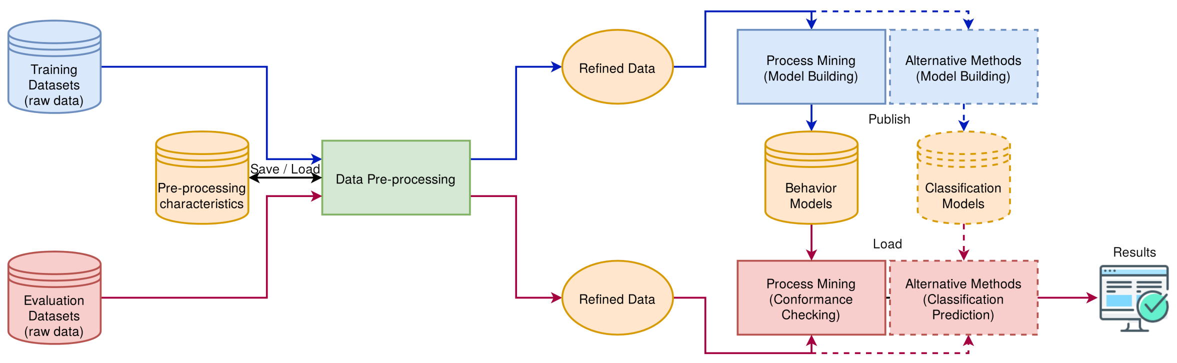

Training Datasets (raw data)
Pre-processing characteristics
Save / Load
Data Pre-processing
Evaluation Datasets (raw data)
Refined Data
Refined Data
Process Mining (Model Building)
Alternative Methods (Model Building)
Publish
Behavior Models
Classification Models
Load
Process Mining (Conformance Checking)
Alternative Methods (Classification Prediction)
Results

Fig. 1. Architecture of our predictive security approach for supporting IoT infrastructures

Section II describes existing work related to IoT security and highlight their limitations. Section III describes and formalizes our predictive security approach based on process mining and data pre-processing for detecting attacks and misbehaviors in IoT environments, as well as the integration of several alternative commonly-used detection methods. Section IV details the implementation prototype, together with extensive series of experiments for evaluating comparatively the benefits and limits of our solution based on multiple criteria. Section V gives conclusions and points out future research work.

II. Related work

Systems and applications based on IoT devices appear to be highly vulnerable to security attacks due to several factors [5]. The complexity of these systems relying on heterogeneous IoT protocols and platforms make security management tasks difficult to implement. In addition, the nature of IoT devices that are characterized by limited resources makes them an attractive target. In that context, they are often affected by naïve weaknesses such as default credentials, poor maintenance and misconfigurations [4], [6], [8].

Solutions have already been largely proposed in the literature to address security issues induced by the Internet-of-Things. Amongst them, [14] describes a set of recommended designs with respect to the architectural building of IoT-based systems. These designs take into account different security, performance and scalability criteria. The security mechanisms that are considered for them depend also on the nature of attacks. External attacks often aim at acquiring the same rights than the ones of authorized internal users. In order to avoid these attacks, the system typically relies on authentication methods based on cryptographic techniques [15]. Internal attacks may be more difficult to counter, and require behavioral patterns to be built, in order to detect deviations that characterize potential security attacks afterwards. There are also works, like in [16], which have proposed an approach to predict the next steps of an attack. It is done by connecting attack graphs, made by experts, and IDS alerts. Thus they detect the current step of the attack and can predict what the next one will be. Applying such method on IoT devices could be a challenge due to their constraints resources. While solutions such as [14] do not introduce a specific method to process and detect misbehaviors, we propose to formalize the different steps of a process mining approach for supporting IoT security in different application domains. Some efforts such as [17] have focused on protecting IoT systems implementing specific routing protocols, in particular the RPL protocol for low-power and lossy networks. Our solution aims at coping with the heterogeneity of IoT protocols and frameworks through the analysis of application data coming from various sources.

Some approaches, such as decision tree learning have been used in order to help detecting misuse in computer network in [18]. It is one of the most intuitive and popular machine learning methods for classification. This method relies on recursively partitioning the data points according to their features values, until a stopping condition is reached. A major limitation of the decision tree method is it builds models that are often overfitting. In other words, the learned model tightly fits the training data. Consequently, slightly inaccurate training data can dramatically infect the model. To overcome this limitation, the random forest method [19] has been proposed. It consists in relying on many decision trees instead of a single one. Each tree is built based on different samples of data as well as different subsets of features. The decision is taken based on a majority vote or a scoring function, which returns an abnormality degree for each observation.

By design, the decision tree method expects to match new data with the ones, that have been used in the model building, sharing the most similarities. Indeed, decision tree and random forest are supervised machine learning methods. However, in our research, we focus on non-supervised detection. To cope with this challenge, we will consider in particular one-class random forest (OCRF), an extension of random forest, which has been proposed for this purpose in [20]. In this paper, the authors have proposed a method relying on decision tree and inspired by random forest that aim to work for unsupervised anomaly detection and have compared it to several well known algorithms, such as isolation forest [21] and support-vector machine techniques. It appears than one-class random forest is the algorithm that shows the best performance in their work and this is why we will mainly use this algorithm as a comparator.

Statistical and analytical methods have already been specified to characterize IoT data and infer potential attacks. For instance, the authors of [22] consider the statistical distribution of the electromagnetic signals generated by IoT devices. In addition, the authors of [23] exploit several machine learning methods applied to datasets generated by smart cities. In the same manner, the authors of [24] identify botnet behaviors from a public NetFlow dataset issued from IoT-based systems. First, they standardize the collected data that are then clustered into two distinct clusters. The cluster containing the highest number of data points is considered as the one characterizing normal behaviors, whereas the one with the smallest number of points is considered as the one characterizing misbehaviors. While this approach contributes to a certain extent to automation, it does not provide significant contextual information to support security analyst experts. In particular, each data point is considered individually, independently from previous states. More elaborated techniques such as Hidden Markov Model (HMM) and neural networks, with long short term memory (LSTM) have been experimented in [25] and [26], respectively. However, the complexity of obtained results makes them difficult to be exploited by security analysts when managing alerts. We rather consider process mining methods in order to generate petri net models that permit to further interpret the different states and behaviors of an IoT-based system.

Process mining (PM) methods have shown their benefits in different areas [27], and are detailed and compared to other data mining techniques in [12]. Typically, they are exploited to detect abnormal sequences of events from specific logs, that may characterize system failures and attack attempts [28]. They are typically used with pre-processing techniques such as clustering and normalization [29]. They may also be combined with machine learning (ML) techniques. In particular, the authors of [30] propose an approach that uses PM techniques to extract the most frequent patterns of an observed industrial system, while ML techniques permit to allocate resources in an optimized manner based on this analysis. A use case dedicated to healthcare with data sensors has also been described in [31], but focusing on detecting anomalies with respect to patients that may reveal diseases. Our purpose is to exploit process mining for supporting the security of IoT-based systems, that are typically distributed, heterogeneous and limited in terms of resources.

III. Predictive Security Approach

We will now describe our process mining approach for supporting IoT predictive security. After giving an overview of the system and components of our solution, we will detail the two main phases respectively related to the building of behavioral models, and to the detection of misbehaviors and potential attacks. The solution is compatible with heterogeneous protocols and platforms that may compose such elaborated systems. Finally, we will characterize the machine learning algorithms that will serve as comparison against our solution.

A. Overview of the system

The system supporting our process mining approach is depicted in Figure 1. It corresponds to a pipeline composed of three main building blocks: a data pre-processing block in front of two other blocks. There are a model building block and a misbehavior detection block for the process mining and the machine learning methods. This pipeline takes as inputs raw data, that may correspond to both training datasets that are used by the model building block, or runtime monitoring datasets that are used for detection purposes based on the behavioral models built from the previous block.

During the model building phase (blue arrows on Figure 1), the data preprocessing block transforms the raw data into refined ones, which are interpretable by the model building block. These refined data are then used by process mining algorithms to generate behavioral models or by machine learning algorithms to build classification models. The behavioral models are formally expressed as petri nets representing discrete event models of the observed system.

During the detection phase (red arrows on Figure 1), the raw data correspond to monitoring data at runtime. They are also transformed by the data pre-processing block. The refined data are then compared by the misbehavior detection block based on the behavioral models, in order to detect potential attacks. The whole pipeline supports different categories of data. While process mining algorithms usually expect event logs, we consider more heterogeneous data inputs following the description below. The considered dataset corresponds to a trace T composed of a set of n records R_i, such as given by Equation 1.

$$T = \{R_1, \ldots, R_n\} \tag{1}$$

Each record R_i of the trace contains m elements E_{ij}, as depicted in Equation 2, while each element E_{ij} consists itself in an attribute/value pair, as given in Equation 3. We can also notice that the different attributes of records R_i are the same in a given trace T, as shown in Equation 4.

$$\forall i \in [\![1;n]\!], R_i = \{E_{i1}, \ldots, E_{im}\} \tag{2}$$

$$\forall i \in [\![1;n]\!], \forall j \in [\![1;m]\!], E_{ij} = \{attribut_{ij} : value_{ij}\} \tag{3}$$

$$Let\ j\ \in [\![1;m]\!], \forall i,k \in [\![1;n]\!], attribut_{ij} = attribut_{kj} \tag{4}$$

In the following of the paper, we will describe the two main phases of our security process mining approach, corresponding to the model building and the misbehavior detection. We will also detail the methods exploited during the data pre-processing, and the considered evaluation metric to quantify misbehaviors in such IoT-based systems.

B. Building of behavioral models

The model building phase consists in generating behavioral models from the raw data of the analyzed system. We have decided to elaborate a solution capable to cope with the heterogeneity of protocols and platforms in an IoT-based system. The phase starts with a data pre-processing block, which transforms the raw data to be interpretable by the process mining algorithms used by the model building block.

Data pre-processing block: the data pre-processing block is composed of three sub-blocks, corresponding to data normalization, data clustering and data splitting. It permits to infer the different states of the observed system. A state can be represented by a tuple of features (F_1, F_2, F_3,...), where two tuples have to be strictly equal to correspond to the same state. However, this approach is inadequate to handle non-categorical and non-boolean features. As a consequence, continuous numerical data are processed by a data normalization and clustering sub-blocks in order to be aggregated into clusters, while the other data (boolean and categorical ones) can directly be used to generate refined datasets. In that context, the continuous numerical data are stored into a dedicated list, noted $L_{continuous}$, while the other data not requiring preliminary treatments are stored into the list noted $L_{\overline{continuous}}$.

During the **data normalization sub-block**, the architecture integrates and re-scales the different features, so that they can be properly compared in the following steps. The data collected from the IoT-based systems represent several features, that are exploited for supporting predictive security. These features may typically be projected into a metric space to quantify the distances amongst data points. However, such quantification is not adequate, when the data are not properly scaled or normalized. By considering the dataset defined by Equation 1, we introduce the Equation 5 formalizing on which data the normalization is applied, while the Equation 6 specifies the outputs of this data normalization sub-block, which in turn serve as inputs for the clustering sub-block. In that context, the element noted EN_{ij} stands for the element E_{ij} associated to its normalized value.

$$\forall i \in [\![1;n]\!], \forall j \in [\![1;m]\!], In_{\text{norm}} = (E_{1j}, \ldots, E_{nj}) \backslash A \quad (5)$$
$$where\ A = \{E_{ij} \,|\, attribut_{ij} \in L_{\overline{continuous}}\}$$
$$Out_{\text{norm}} = (EN_{1j}, \ldots, EN_{nj}) \backslash B \quad (6)$$
$$where\ B = \{EN_{ij} \,|\, attribut_{ij} \in L_{\overline{continuous}}\}$$

The normalization parameters are stored into the pre-processing characteristics database, so that they can be exploited during the detection phase. This enables maintaining the consistency of normalization parameters during these two main phases.

During the **clustering sub-block**, the refined data are first aggregated into clusters that serve to define the system states. Clustering techniques are commonly used in the area of data mining. In our context, they permit to reduce the number of states that characterize the IoT-based system. Each tuple of continuous numerical elements specified in Equation 6 will be associated to a single cluster identifier, noted Cl_i, as given by Equation 7. The properties of clusters, such as the barycenters and the maximal distance between normalized data and these barycenters inside a given cluster, are stored and exploited during the detection phase.

$$\forall i \in [\![1;n]\!], \forall j \in [\![1;m]\!], Out_{\text{cl}} = (E_{i1}, \ldots, E_{im}, Cl_i) \backslash C (7)$$
$$where\ C = \{E_{ij} \,|\, attribut_{ij} \in L_{continuous}\}$$

The tuples, called $Out_{clustering}$, described by Equation 7, permit to define the different states of the system. Let us consider that the system states are identified by a state identifier, noted S_p with $p \leq n$. When two tuples are stricly characterized by the same values, then they are describing the same state, and are therefore identified by the same state identifier S_p. As a consequence, the records R_i of the trace T can be reduced to two elements: a timestamp and a state identifier, as depicted by Equation 8.

$$R_i = \{timestamp : t_i, state : S_p\} \quad (8)$$

During the **splitting sub-block**, the records of the trace are split into k data subsets that are noted P_l and correspond to different time intervals of the same duration. This splitting mechanism aims at reducing the complexity of behavioral models, by preventing a too high characterization that may prevent a proper detection of misbehaviors and attacks.

$$\forall a, b \in [\![1;n-1]\!]\ with\ a \geq b\ and\ l \in [\![1;k]\!],$$
$$P_l = \begin{cases} R_a \\ \vdots & \text{with } t_b\text{-}t_a \leq I \\ & \text{but } t_{b+1}\text{-}t_a > I \\ R_b \end{cases} \quad (9)$$

Process mining block: the process mining algorithms generate behavioral models from the different subsets that are defined in Equation 9. These models are then exploited during the detection phase to identify deviations. In particular, when none of the models built from the subsets is close to the monitoring data collected at runtime, the IoT-based system will be considered as misbehaving. Amongst processing mining techniques, we have decided to focus on the inductive mining algorithm, which is capable of efficiently support the building of behavioral models that perfectly match event logs. This algorithm, described in [32], is organized into three main steps. First, it establishes a directly-follow graph, which is a graph summarizing the occurrence of events in the event logs, corresponding in our case to the refined data. It then infers a process tree from this graph. To do so, the algorithm looks for the most adequate operators (exclusive choice, loop, parallelisation) in order to cut the directly-follow graph. Finally, the algorithm infers from the obtained process tree, a petri net characterizing the behavior of the IoT-based system. Therefore, each subset, defined by Equation 9, leads to a petri net, noted M_l.

C. Detection of misbehaviors

We will now describe the second phase corresponding to the detection of misbehaviors. The approach consists in analysing monitoring data at runtime, and comparing them to the behavioral models built for the IoT-based system. We will first introduce the metric considered to quantify the deviation from these models, and then detail the detection mechanism.

Deviation quantification: it is important to quantify the deviation (or alignment) of the refined data with the behavioral models obtained from the model building phase. This is required to detect potential misbehaviors, but also to evaluate the performance of the generated models. Let us consider a behavioral model and a refined dataset to be evaluated. First of all, the model and the dataset have to be aligned using a dedicated method. This alignment method can be described as follows. For each event from the dataset, when the same movement (i.e. changing to one state to another one) can be performed on both the behavioral model and the considered log, then this event is considered as synchronized. A movement cost is equal to 0, when the model and its log are synchronized, otherwise it is equal to 1. The alignment cost is obtained by summing the different movement costs.

In that context, we consider the fitness metric detailed in Equation 10, to evaluate whether the considered model, noted M, can replay a given trace or log, noted T, in an accurate manner. The closer this metric is to 1, the more the model is capable to replay the given log.

$$Fitness_M(T) = 1 - \frac{Cost(M,T)}{Move(T) + Len(T) \times Move(M)} \tag{10}$$

In this equation, $Cost(M,T)$ stands for the optimal alignment cost between M and T, while $Move(T)$ corresponds to the total cost of desynchronized movements on the log. $Move(M)$ corresponds to the same total cost for the model, while $Len(T)$ indicates the number of states in the log. The denominator of the formula represents the maximum possible value of the total alignment cost, when there is not a single synchronized movement between the log T and the model M in the optimal alignment.

Detection mechanism: the misbehavior detection is preceded by a data pre-processing. It consists in generating and formatting sub-logs corresponding to records composed of timestamps and state identifiers. The same normalization parameters are applied, and the clustering sub-block is restricted to a binding mechanism, during which continuous numerical data are associated to the closest existing cluster, as long as the distance between the cluster and the data point does not exceed the maximal distance found during the model building phase. This enables the mapping of the new monitoring data to the previously obtained clusters. The dataset is then split into smaller subsets corresponding to time intervals of the same duration, as previously described. These subsets are then replayed with the behavioral models generated for the IoT-based system. The objective is to find for each subset the corresponding behavioral model, i.e. the model for which the fitness is higher. The closer the fitness is to 1, the more the behavioral model is capable to replay the given subset.

To better formalize the detection mechanism, let us consider l data subsets P_i, coming from monitoring data at runtime, and m behavioral models M_i generated during the model building phase. The approach consists in finding the model that best fits each data subset P_i, i.e. the model characterizing the highest fitness, as given by Equation 11. When a data subset does not fit any behavioral model, then the IoT-based system is considered as misbehaving.

$$\forall i \in [\![1;n]\!], Fitness_i = \mathbf{max}_{j \in [\![1;m]\!]}\, Fitness_{M_j}(P_i) \tag{11}$$

At this stage, we first associate each data subset to the closest behavioral model with the corresponding fitness, this corresponding to the RES_i tuples defined by Equation 12.

$$\forall i \in [\![1;n]\!], \exists j \in [\![1;m]\!], RES_i = (P_i, M_j, Fitness_i) \tag{12}$$

Once these evaluation results RES_i are available, we define a set of q different fitness thresholds FT_r. These thresholds have been considered during our experiments. They permit to distribute the results RES_i into two distinct sets $NORMAL_r$ and $ABNORMAL_r$, defined by Equations 13 and 14. The presence of one single tuple RES_i is enough to characterize a misbehaving IoT-based system.

$$\forall i \in [\![1;n]\!], \forall r \in [\![1,q]\!],$$
$$NORMAL_r = \{RES_i \mid fitness_i > FT_r\}, \tag{13}$$
$$ABNORMAL_r = \{RES_i \mid fitness_i \leq FT_r\} \tag{14}$$

The detection mechanism is compatible with the heterogeneity of protocols and platforms that can compose such systems, and therefore, can support cross-protocol and cross-platform attacks targeting these systems.

D. Alternative detection methods

Different methods have been proposed for anomaly detection. In [33], they are classified into four main categories, namely probabilistic and statistical, linear, proximity-based and high-dimensional. For each of these categories, we consider one of the most used algorithms. They are respectively: elliptic envelope [34], one class SVM [35], local outlier factor [36] and isolation forest [21]. We also consider one class random forest [20], which has been recently proposed for anomaly detection. In the following, we show how these methods can be integrated into the proposed processing pipeline.

a) Elliptic Envelope [34]: The methods belonging to the probabilistic and statistical category assume that data follow a known probability distribution whose parameters are determined during the training phase. In particular, the elliptic envelope assumes an underlying Gaussian distribution. It fits a boundary ellipse to the central data points and considers the outsiders to be anomalous. To estimate the size of the ellipse, this method uses the FAST-minimum covariance determinant [37]. The latter uses the Mahalanobis distance D_M as a scoring function S that assesses the abnormality of a record R_i ($R_i = \{E_{ij} = \{att_{ij} : val_{ij}\}; \forall j \in [\![1;m]\!]\}; \forall i \in [\![1;n]\!]$) :

$$S(R_i) = D_M(R_i) = \sqrt{(val_i - \mu)^T \times S^{-1} \times (val_i - \mu)} \tag{15}$$

where $\mu = (\mu_1, ..., \mu_m)^T$, μ_j is the mean of $\{val_{1j}, ..., val_{nj}\}; \forall j \in [\![1;m]\!]$, $val_i = (val_{i1}, ..., val_{im})^T; \forall i \in [\![1;n]\!]$ and S is the covariance matrix of $(val_i)_{i \in [\![1;n]\!]}$.

b) One Class SVM [35]: Linear methods aim to embed the maximum of data points in a subspace having a dimension lower than the features space. The points that do not fit the embedding are considered anomalous.

One class SVM has been deigned according to this principle. Its main feature is adding margins to the embedding subspace to limit overfitting. To classify a record R_i (i.e., a data point), this method relies on the following scoring function $S(R_i)$:

$$S(R_i) = \sum_{j=1}^{m} \alpha_j . K(R_i, val_{ij}) - \rho \tag{16}$$

where $(\alpha_j)_{j \in [\![1;m]\!]}$ are the Lagrange multipliers, K is the kernel function that projects records into the embedding subspace and ρ is the margin. Lagrange multipliers are calculated by optimising:

$$\forall i \in [\![1;n]\!], \forall j,k \in [\![1;m]\!] : \min_{\alpha} \frac{1}{2}\alpha_j.\alpha_k.K(val_{ij}, val_{ik}) \tag{17}$$

such as $0 \leq \alpha_j \leq \frac{1}{\nu m}$ and $\sum_{j=1}^{m} \alpha_j = 1$ where ν is the ratio of records expected to be anomalous.

c) Local Outlier Factor [36]: Proximity-based methods consider a data point as anomalous if its proximity (i.e., locality) is sparsely populated. The local outlier factor is a score S of abnormality. It does not only consider the distances between data points but also the local densities. For a record R_i, it corresponds to the ratios average of the local reachability density (lrd) of R_i and that of R_i's k-nearest neighbours:

$$S(R_i) = lof(R_i) = \frac{1}{k}. \sum_{R_j \in knn(R_i)} \frac{lrd(R_j)}{lrd(R_i)} \tag{18}$$

where $lrd(R_i) = \frac{k}{\sum\limits_{R_j \in knn(R_i)} reach-dist(R_i,R_j)}$, k is a parameter, $reach\text{-}dist(R_i, R_j) = max\{k - dist(R_j), d(R_i, R_j)\}$, $k - dist(R_j)$ is the Euclidean distance of k-neighbors nearst to R_j, $d(R_i, R_j) = \sqrt{\sum_{j=1}^{m} |E_{ij} - E_{ij}|^2}$, $knn(R_i)$ is the set of the k nearst records to R_i.

d) Isolation Forest [21]: Identifying anomaly in data having high dimension is particularly challenging because irrelevant dimensions may mask anomalies. Isolation forest is one of the most efficient methods in handling the "curse of dimensionality". It randomly splits data into subsets having reduced dimensions and builds a binary tree for each subset. The tree is obtained by recursively splitting the data subset based on a randomly selected attribute att_{ij} and val_{ij} until a maxium tree height is reached or only one record remains in the subset or all the remaining records have the same values. To assess the abnormality of a record R_i, it computes $S(R_i)$, the average depth of the leaves to which R_i is associated:

$$S(R_i) = 2^{-\frac{E(h(R_i))}{c(n)}} \tag{19}$$

assuming $c(n) = 2H(n-1) - 2(n-1)/|F|$, $H(i)$ is the i^{th} harmonic number: $H(i) = \sum_{k=1}^{i} \frac{1}{k}$.

e) One Class Random Forest [20]: One class random forest is an extension of the random forest method. To deal with the absence of abnormal data during the training phase, these data is considered to be tightly concentrated around normal one. More specifically, in each node o of a built tree, the abnormal data is updated so that its cardinal becomes proportional to the one of normal data. Let F_o denote a subset of the trace T in the node o. Thus:

$$|F'_o| = \gamma |F_o| \tag{20}$$

where γ is typically set to 1. To assess data heterogeneity, the Gini index is used. It is expressed this way in a two-class classification problem (such as anomaly detection):

$$i_G(o) = 2\frac{|F_o|}{|F_o| + |F'_o|}\frac{|F'_o|}{|F_o| + |F'_o|} \tag{21}$$

where $|F_o|$ (resp. $|F'_o|$) stands for the number of observations with label 0 (resp.1) in node o. To decide about anomaly of a record $R_i \in F$, the scoring function of random forest is adapted as follows:

$$log_2(S(R_i)) = -(\sum_{o \in leaves} \mathbb{1}_{R \in o} d_n + c(|F_o|))/c(|F|), \tag{22}$$

where d_o is the depth of node o, c(|F|) = 2H(|F| - 1) - 2(|F| - 1)/|F|, $H(k)$ is the k^{th} harmonic number: $H(k) = \sum_{l=1}^{k} \frac{1}{l}$.

IV. Prototype and Performance Evaluation

We have developed a proof-of-concept prototype implementing our solution, and have performed extensive series of experiments, in order to evaluate and compare the performance of process mining to alternative commonly-used detection methods, including elliptic envelope, support-vector machine, local outlier factor and isolation forest techniques. In particular we have considered the influence of the time splitting, the influence of the preprocessing clustering and the detection performance according to multiple criteria.

A. Experimental setup

The proof-of-concept prototype follows the architecture of our predictive security approach for IoT infrastructures, and includes the two main building blocks supporting the process mining activities. The first block is responsible for building behavioral models according to different configuration parameters (such as normalization and clustering). The second block is responsible for evaluating the misbehavior detection and its performance. The data pre-processing is developed in Python 3.6, whereas the ProM library is used for applying process mining algorithms and for evaluating datasets with respect to behavioral models. We consider the version 6.8 of the ProM framework environment which has been developed under Java 8. The process pipeline is integrated into a docker container. All the experiments have been performed over a 3.3 Ghz Intel Core i5 4th-generation computer with 16 GB of RAM memory.

We use Elasticsearch to store the training and evaluation datasets. This framework consists in an open source search engine, that is designed to be scalable, distributive and near real-time. It is widely used in big data environments thanks to its convenience to the most used architectures, such as Lambda and Kappa solutions. In order to store the behavioral models and the pipeline configurations, we consider a MongoDB database, and use Apache Kafka to manage alarms about anomalies. These alarms may be consumed from Kafka by a dashboard or by any application able to select and apply actions and counter-measures in regards to the detected anomaly.

In our experiments, we have considered three main datasets [38] from different elaborated IoT-based systems, including connected cars, industry 4.0 and socially-assistive robots. These datasets were provided by our partners in the European H2020 SecureIoT project. Table I provides an overview of them. The first dataset corresponds to a connected cars use

TABLE I
Experimental Dataset Overview

	Connected Cars	Industry 4.0	Assistive Robots
Data Rate	2 Hz	2 Hz	20 Hz
Model Building (nb of samples)	3200	9000	12000
Evaluation (nb of samples)	2400	7000	10000
Anomaly Ratio	20%	30%	10%

case and contains sensor data, such as speeds, steering angles, currently engaged gears, and engine rotation, that are generated by an industrial driving simulator reproducing the behavior of drivers. The considered attacks are man-in-the-middle attacks that may impact on the overall car control. The second dataset corresponds to the case of industry 4.0, with sensor data generated by an industrial plastic molding system. The system supports the production of plastics pieces based on an injection molding process. The available data gives information related to the temperatures at several locations in the system, as well as the pressure inside every relevant pistons within the process. The considered attacks are intrusions enabling to send malicious commands to the process, and may lead to a machine breakdown. The third dataset corresponds to data generated by socially-assistive robots, that support autistic children to learn and increase their social skills using exercises and interactions. The dataset includes physical and application robot information data, such as its motor positions. The considered attacks are intrusion attacks over the robot operating systems, that permit again to take control on the robot behaviors, with potential physical damages. As shown in next sections, we have evaluated the influence of the time splitting, the influence of the pre-processing clustering, and the detection performance according to multiple criteria in comparison to several commonly-used detection methods, including elliptic envelope, support-vector machine, local outlier factor, isolation forest and one class random forest techniques.

B. *Influence of the time splitting*

In a first series of experiments we wanted to quantify the influence of time splitting on the performances of our solution. Time splitting occurs during data pre-processing, and is characterized by a time interval at which the dataset is split into smaller subsets over time. The benefits of such splitting is two-fold. It permits to minimize the complexity of built behavioral models, and enables a more precise identification of the data corresponding to a misbehavior or a potential attack. However, it may also degrade the overall detection performances. Figure 2 describes the receiver operating characteristic (ROC) curves obtained with the three experimental datasets, while varying the time interval parameter used for splitting data. Sub-figures 2 (a), (b), (c) correspond respectively to the connected cars, industry 4.0 and assistive robot datasets. These curves characterize the detection performances in terms of false positive rate (x axis) and true positive rate (y axis). The area under the curve (AUC) permits to quantify the performance for a given time splitting, with higher AUC values corresponding to better performances.

TABLE II
Best Experimental Results for each Dataset

	Connected Cars	Industry 4.0	Assistive Robots
BIRCH	0.824	0.5	0.818
K-means	0.979	0	0.848
DBSCAN	0.694	1	0.5

For each of the three datasets, we can observe the most adequate time splitting (highest AUC value), while varying the time interval (TS parameter). With the first dataset, we obtain the best AUC value of 0.923 with a time interval of 10 seconds. From such an optimal configuration, the detection performance is dropping when we increase or decrease this TS parameter with the three scenarios. In particular, when the time interval is too low, there is a high risk to obtain subsets with only a few data points, which is not adequate for replaying the traces on the built behavioral models. In such cases, the process mining does not bring any additional value in comparison to a clustering-only method. It actually brings a higher complexity for the same detection performance. Considering a too high time interval is also an issue due to the complexity of models. Besides, the identification of misbehaviors and attacks may be less efficient during the detection phase. We can also observe this phenomenon when measuring the processing time during the model building and detection phases. This processing time decreases during the model building phase with the highest TS time intervals, as less models are required to be generated. During the detection phase, the observed processing time is the highest with the two extreme cases, corresponding to the highest TS value and the lowest TS value, while it appears to be minimal with the best AUC value.

C. *Influence of the preprocessing clustering*

In these experiments, we focused on assessing the impact of the preprocessing clustering phase. We have varied the different clustering techniques (K-means, BIRCH, DBSCAN [39]) and the normalization techniques (min-max, z-score). We ensure that the parameters are the same between the building phase and the corresponding detection phase. Modifying the parameters of the clustering algorithm has no significant impact on the processing time, with a model building time of around 30 seconds, but influences the detection performance. Table II reports the values of the area under the ROC curves corresponding to the three datasets, with the best results for each of the three clustering techniques mentioned before. We notice than the best experimental results obtained respectively with the connected car and assistive robot datasets, are found considering the K-means clustering combined with min-max normalization, while with the industry 4.0 dataset, the best result is obtained with DBSCAN clustering combined with min-max normalization.

Assuming K-means clustering, we varied the k parameter setting the number of clusters, which so changes the number of states that are then considered by the process mining. With DBSCAN, we varied the epsilon (eps) parameter that specifies the local radius for expanding clusters. Finally, with BIRCH, we varied the threshold parameter that specifies the

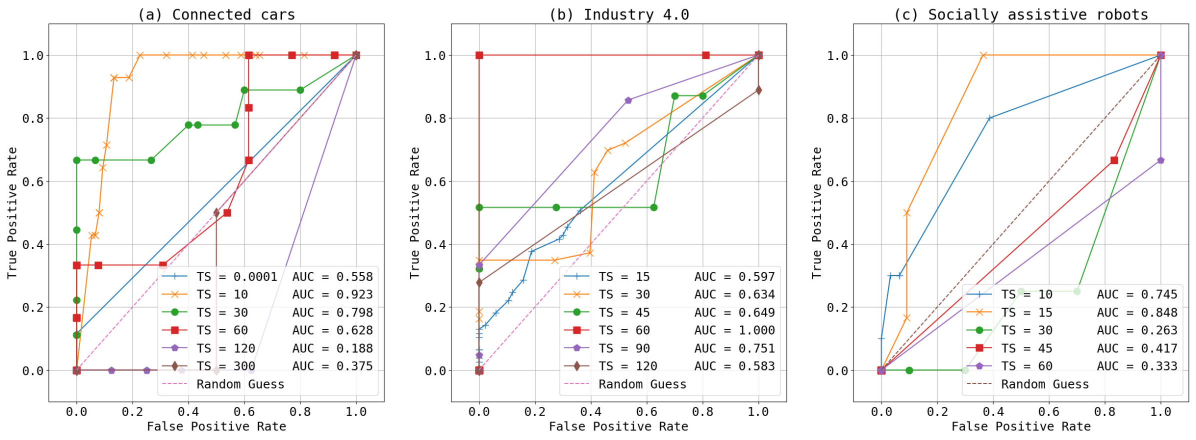


Fig. 2. Influence of the time splitting on the detection performance based on ROC curves

radius of the clusters. Increasing the k parameter with K-means clustering, decreasing the eps parameter with DBSCAN clustering or decreasing the threshold with BIRCH clustering permits to improve the characterization of misbehaviors and attacks with a large set of states. However, we can notice that considering a high number of states may also degrade the detection performance. Indeed, this leads to generate too specific behavioral models that are unable to properly capture deviations from the learnt model. At the extreme case, each point of a given dataset could be considered as a unique cluster, and therefore a unique state. This lack of data clustering prior to the process mining leads so to an explosion of number of states and so to low performance. For instance, with the first dataset, the case with the k parameter set to 2500 induces a ROC curve close to the diagonal axis, with an Area Under the Curve of around 0.5, characterizing bad detection results (*e.g.* similar to a random decision). These different experiments argue in favour of our predictive security solution combining process mining with clustering techniques.

D. *Influence of noisy and missing data*

In order to evaluate the robustness of our approach, we have considered the influence of noisy and missing data on the detection performance. The evaluation datasets have been modified by either injecting a Gaussian noise, or by randomly removing a discontinuous portion of the data. We have considered the pre-processing methods that provided the best performance for the three industrial datasets. We have then compared the Matthew Correlation Coefficient (MCC) between our process mining method and the isolation forest method. This coefficient is one of the best performance criteria to describe the confusion matrix. The results given by the alternative methods described in Section III-D were close in these experiments. We therefore focus on one of them, the isolation forest method, to describe the influence of noisy or missing data on the detection.

Figure 3 describes the detection performance for the process mining and isolation forest methods, while varying the percentage of noise. It appears that the process mining approach is more sensitive to noisy data than the isolation forest approach. However, it is still robust for the simulated datasets

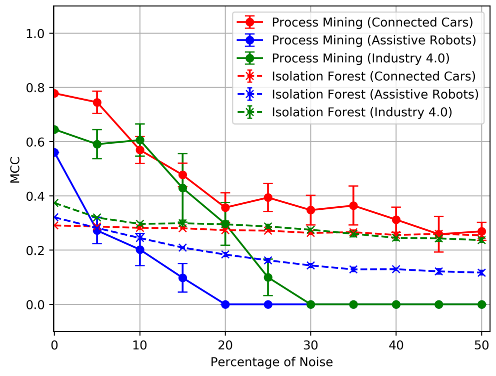


Fig. 3. Influence of noisy data on the detection performance based on MCC

(connected cars and industry 4.0) and performs better than the alternative methods, when the noise is of less than 20%. Regarding the socially-assistive robot dataset, an immediate decrease of detection performance has been observed during the experiments. This decreased performance is mostly due to the noise generated by the system. Moreover, most of the data points defining the various clusters are close to each other, this is why adding noise to the robot dataset can easily change the cluster of a data point.

Figure 4 shows the performance of the methods with missing data. Our approach performs better than the alternative methods, when the portion of the missing data is less than 25%. However, for a higher threshold, the performance decreases significantly for the connected car dataset and the socially-assistive robot dataset. On the contrary, the performance of the industry 4.0 dataset stays stable even with 50% of missing data. It is mostly due to the anomaly ratio, that is higher for the industry 4.0 dataset, as shown in Table I, and the duration of the anomalies that are more important for this specific dataset. However, its performance decreases as well when using a higher missing data ratio. These experiments show that our process mining approach can partially address the issues induced by noisy and missing data: the normalization and clustering phases that are performed prior to the process mining permit to reduce the influence of the noise, while the data rate can be adjusted to minimize the impact of missing data.

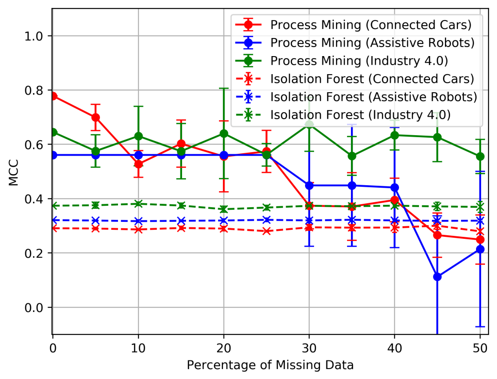


Fig. 4. Influence of missing data on the detection performance based on MCC

E. Comparison with alternative detection methods

In a third series of experiments, we compared the detection performance of process mining to the five previously-mentioned alternative methods: elliptic envelope, local outlier factor, support-vector machine, isolation forest and one class random forest. We evaluated each of them with the three industrial datasets. For the one class random forest, we used the implementation proposed by this method's authors while for the other detection methods, we used the implementations available in the Scikit-learn library [40]. To maximize the detection performance, we tuned the implementation parameters, and pre-processed data with min-max and z-score normalizations when appropriate. In particular, for process mining, the pre-processing included also testing different clustering (K-means, BIRCH, DBSCAN) and time splitting techniques. Different metrics have already been proposed for the evaluation of such detection methods, but none is sufficient in itself [41]. For this reason, we considered several of them that are presented in detail in [41] and [42].

In Figure 5, we plot the Receiver Operating Characteristic (ROC) curves, which represents the True Positive Rate (TPR) against the False Positive Rate (FPR) realized by each detection method at various thresholds. We also plot the Precision Recall (PR) curves to represent the precision against the recall, in Figure 6. Using the score functions 15, 16, 18, 19, and 22 as well as the fitness 10, we select the thresholds that maximize the TPR and minimizes the FPR on the ROC curves. Then, we measure the precision, recall, accuracy, ROC's Area Under the Curve (ROC AUC), PR's Area Under the Curve (PR AUC), F1 score (i.e., the harmonic mean of precision and recall), Matthews Correlation Coefficient (MCC) (i.e., the correlation between true and predicted values) and the Log Loss (LL) (i.e., the uncertainty of the model). The obtained results are depicted in Figure 7, corresponding to the three experimental datasets. Since one class random forest, isolation forest and elliptic envelope are randomized algorithms, we considered the means of 30 experiments according to the rule of thumb. We neglected the variances of these methods because they did not exceed 6% of the mean and were even less than 1% in most cases. Exceptionally, however, the log loss of the socially assistive robot analysis with one class random forest varied significantly with 27% from the mean (this observation will be discussed in the following paragraphs). Figure 5 shows that process mining has the best ROC curves for the three datasets. Indeed, this method curves have the best TPR and FPR ratios, which are represented by the closest points to the (1,0) one. In addition, compared to the others, this method realised AUC ROCs that were at least 12%, 33% and 9% higher in the connected cars, Industry 4.0 and socially assistive robots, respectively.

Figure 6 shows that process mining has also the best PR curves for the three datasets. Indeed, this method realised PR ROCs that were at least 12%, 33% and 9% higher in the connected cars, Industry 4.0 and socially assistive robots, respectively. Figure 7 depicts the performance of the detection methods in analysing each dataset. In the case of the (a) connected cars and (b) industry 4.0, process mining out performs the other detection methods, since it achieved better values for each metric. However, this was not the case of the (c) socially assistive robots' dataset, since the one class SVM, one class random forest and isolation forest achieved the best values for accuracy and log loss. As a matter of fact, this dataset is the most imbalanced: as depicted in Table I, the misbehavior ratio of this dataset is 10% only, while it is 20% and 30% for the connected cars and industry 4.0 datasets, respectively. Consequently, this dataset is the most vulnerable to the accuracy paradox, where detection methods can achieve high accuracy by over-classifying the data points as non-anomalous. Obviously, this was the behavior of these alternative methods because the better are their accuracy and log loss, the worse is their recall (i.e the portion of anomalous data points that were identified). Contrary, to the accuracy and log loss the MCC considers the size of the four entries of the confusion matrix. Consequently, it is more suitable to imbalanced datasets. Figure 7 (b) shows that process mining achieved the highest values of this metric. More specifically, its values were 47%, 79% and 62% higher than those of the one class SVM, one class random forest and isolation forest, respectively. For these reasons, we believe that process mining is more suitable than those methods to analyze the socially assistive robots' dataset. This method is also more suitable than the elliptic envelope and the local outlier factor. Even though the latter achieves an interesting recall (just 0.4% lower than the one of process mining), the remaining values it achieves are significantly lower than those of process mining.

Thus, our method, based on process mining with clustering, has the highest performance in all these experimental scenarios. It is worth mentioning that we found the most accurate results for this method, when the time splitting is parameterized in a similar manner (i.e. using the same duration for the time intervals) during the model building and the misbehavior detection phases. As it is suggested above, the data pre-processing is a decisive block and its purpose is to bring out the different states of a studied system. For datasets giving application data of evolutive systems that can be modeled, the results we have obtained support the relevance of our approach that takes into account the previous states. Thus, using process mining has given better results than the other commonly used detection methods. Using the Principal Component Analysis [43] technique, we could reduce the

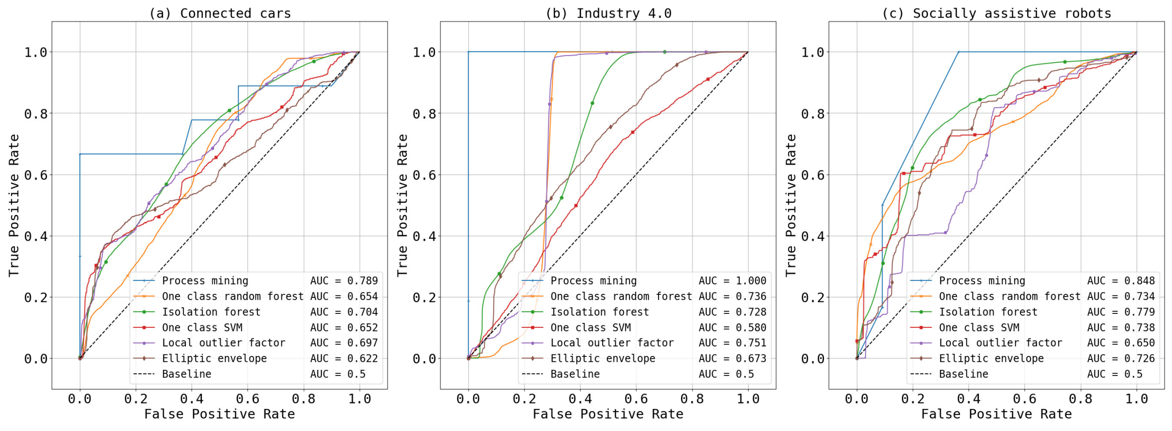


Fig. 5. Comparison based on ROC curves of process mining to other detection methods

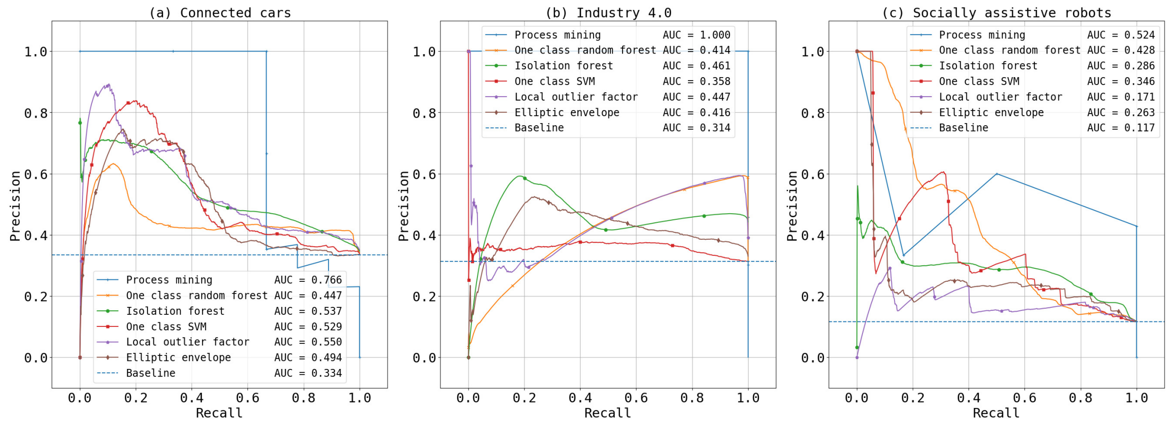


Fig. 6. Comparison based on PR curves of process mining to other detection methods

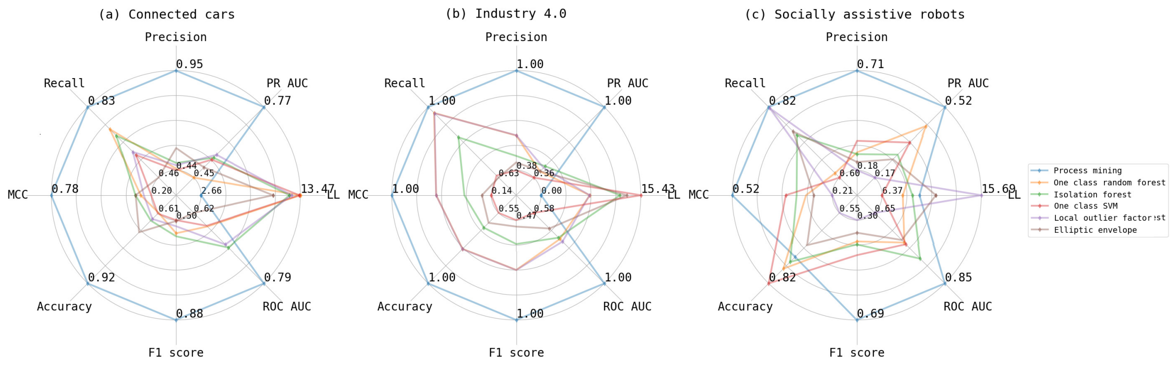


Fig. 7. Multi-criteria performance comparison of process mining to other detection methods

dimensions of the datasets and visualize them. We noticed that the anomalous data points are distributed differently. In the connected cars dataset, they were rather scattered. This explains why the alternative detection methods performed rather modestly in this dataset analysis. For the socially assistive robots, the abnormal data points were rather clustered. For this reason, the LL of the one class random forest varied significantly and some alternative detection methods could achieve high accuracy, recall and low LL. The anomalous points in the industry 4.0 dataset are tightly concentrated around normal ones. Since one class random forest makes this assumption, it was relatively efficient in analyzing this dataset. Local outlier factor was also as efficient as one class random forest. This is due to the fact that this algorithm considers anomalous points locally instead of building a space gathering only normal points as done by the one class SVM, which had relatively low performance in this context. The elliptic envelope had also low performance because it assumes that anomalous points are far from normal ones. As a matter of fact, our approach requires more time to process the raw data. Typically, it requires between 19 and 55 seconds for model building and evaluation, while isolation forest requires less than one second time for these tasks on average. This time difference is due to the clustering pre-processing, which is a requirement for the process mining approach. Since this limit was not investigated in this work, we envision to study it in depth in the future. To cope with it, it is possible to rely on a hybrid solution that uses one of the alternative detection methods in addition to our approach.

V. Conclusions

Security management is a key challenge for the protection of elaborated IoT-based systems, that are often characterized by heterogeneous protocols and platforms, and that integrate devices with limited resources (memory, CPU, battery). In this context, we have evaluated the exploitability and performance of process mining to cope with a large variety of devices and protocols, for supporting IoT predictive security. After describing the underlying architecture, we have detailed the different phases related to this solution, from the building of behavioral models to the detection of misbehaviors and potential attacks. We have formalized and integrated into this architecture four other categories of commonly-used detection methods, including elliptic envelope, support-vector machine, local outlier factor and isolation forest techniques, that serve as a support to our comparative analysis. We have also developed a proof-of-concept prototype provided as a docker container, that implements the proposed solution and exploits the ProM library. We have experimented it with three different industrial datasets, such as connected cars, industry 4.0, and robot networks. We have performed extensive sets of experiments in order to evaluate the performance of our process mining approach, and compare it to the four other alternative detection methods, by considering the influence of the time splitting, the influence of the clustering techniques, and the overall detection performance according to multiple criteria. Moreover, we have tested how our approach reacts against noisy or missing data. These results have been confronted to the ones obtained by the isolation forest detection method. The experiments have clearly shown the benefits of combining process mining and data pre-processing, in particular clustering techniques. The data pre-processing permits to identify and minimize the states characterizing the IoT-based system, so that the process mining is not facing a state explosion. The process mining then permits to elaborate behavioral models that are compatible with the heterogeneity of protocols and devices. These behavioral models are then exploited to analyze monitoring data at runtime, and to properly detect misbehaviors and potential attacks in these environments.

As future work, we are interested in evaluating to what extent the generated alerts can be exploited to drive the activation of specific counter-measures in an automated manner. We would also like to investigate hybrid detection techniques, in particular in the case of elaborated and advanced persistent attacks in IoT infrastructures. Finally, we are planning to consider incremental model building methods, so that the behavioral models can be enriched at runtime, in order to further improve the performance of such an IoT predictive security solution.

Acknowledgment

This work has been partially supported by the SecureIoT project, funded by the European Union's Horizon 2020 research and innovation programme under grant agreement no. 779899; the exploited datasets have been provided by IDIADA UK, it's OWL, which has worked with FUJITSU, and LuxAI.

References

[1] M. Alaa, A. A. Zaidan, B. B. Zaidan, M. Talal, and M. L. M. Kiah, "A review of smart home applications based on internet of things," *Journal of Network and Computer Applications*, vol. 97, pp. 48–65, 2017.

[2] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0," *IEEE industrial electronics magazine*, vol. 11, no. 1, pp. 17–27, 2017.

[3] K. Delaney and E. Levy, "Connected Futures Cisco Research: IoT Value: Challenges, Breakthroughs, and Best Practices." Cisco System Report, May 2017.

[4] M. Antonakakis, T. April, M. Bailey, M. Bernhard *et al.*, "Understanding the Mirai Botnet," in *Proceedings of the USENIX Security Symposium*, 2017, pp. 1092–1110.

[5] E. Bertino and N. Islam, "Botnets and Internet of Things Security," *Computer*, vol. 50, no. 02, pp. 76–79, Feb 2017.

[6] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and Other Botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[7] Vitaly Simonovich, "Imperva blocks our largest ddos l7/brute force attack ever (peaking at 292,000 rps)," https://www.imperva.com/blog/imperva-blocks-our-largest-ddos-l7-brute-force-attack-ever-peaking-at-292000-rps/, last visited on September 2020.

[8] L. Rouch, J. François, F. Beck, and A. Lahmadi, "A Universal Controller to Take Over a Z-Wave Network," in *Proceedings of Black Hat Europe*, 2017, pp. 1–9.

[9] Z. Zhang, M. C. Y. Cho, C. Wang, C. Hsu *et al.*, "IoT Security: Ongoing Challenges and Research Opportunities," in *Proceedings of the 7th International Conference on Service-Oriented Computing and Applications*, Nov 2014, pp. 230–234.

[10] B. Thuraisingham, M. Kantarcioglu, K. Hamlen, L. Khan *et al.*, "A Data Driven Approach for the Science of Cyber Security: Challenges and Directions," in *Proceedings of the 17th International Conference on Information Reuse and Integration*, July 2016, pp. 1–10.

[11] A. Hemmer, R. Badonnel, and I. Chrisment, "A Process Mining Approach for Supporting IoT Predictive Security," in *Proceedings of the Network Operations and Management Symposium*, April 2020.

[12] W. Aalst, van der, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Germany: Springer, 2011.

[13] J. B. Fraley and J. Cannady, "The Promise of Machine Learning in Cybersecurity," in *Proceedings of the IEEE SoutheastCon Conference*, March 2017, pp. 1–6.

[14] A. Bassi, M. Bauer, M. Fiedler, T. Kramp *et al.*, *Enabling Things to Talk: Designing IoT Solutions with the IoT Architectural Reference Model*. Springer Publishing Company, Incorporated, 2013.

[15] M. Pahl and L. Donini, "Securing IoT Microservices with Certificates," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium*, April 2018, pp. 1–5.

[16] P. Holgado, V. A. Villagrá, and L. Vázquez, "Real-Time Multistep Attack Prediction Based on Hidden Markov Models," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 1, pp. 134–147, Jan 2020.

[17] A. Mayzaud, R. Badonnel, and I. Chrisment, "A Taxonomy of Attacks in RPL-based Internet of Things," *International Journal of Network Security*, vol. 18, no. 3, pp. 459–473, May 2016.

[18] O. Depren, M. Topallar, E. Anarim, and M. K. Ciliz, "An Intelligent Intrusion Detection System (IDS) for Anomaly and Misuse Detection in Computer Networks," *Expert systems with Applications*, vol. 29, no. 4, pp. 713–722, 2005.

[19] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[20] N. Goix, N. Drougard, R. Brault, and M. Chiapino, "One Class Splitting Criteria for Random Forests," in *Proceedings of the 9th Asian Conference on Machine Learning*, M.-L. Zhang and Y.-K. Noh, Eds., vol. 77. PMLR, Nov 2017, pp. 343–358.

[21] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation Forest," in *Proceedings of the 8th IEEE International Conference on Data Mining*, Dec 2008, pp. 413–422.

[22] N. Sehatbakhsh, M. Alam, A. Nazari, A. Zajic, and M. Prvulovic, "Syndrome: Spectral analysis for anomaly detection on medical iot and embedded devices," in *2018 IEEE international symposium on hardware oriented security and trust*. IEEE, 2018, pp. 1–8.

[23] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi *et al.*, "Machine Learning for Internet of Things Data Analysis: a Survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161 – 175, 2018.

[24] D. S. Terzi, R. Terzi, and S. Sagiroglu, "Big Data Analytics for Network Anomaly Detection from Netflow Data," in *Proceedings of the International Conference on Computer Science and Engineering*, Oct 2017, pp. 592–597.

[25] S. Ramapatruni, S. N. Narayanan, S. Mittal, A. Joshi, and K. Joshi, "Anomaly detection models for smart home security," in *2019 IEEE 5th Intl Conference on Big Data Security on Cloud*. IEEE, 2019, pp. 19–24.

[26] S. Chauhan and L. Vig, "Anomaly Detection in ECG Time Signals via Deep Long Short-term Memory Networks," in *Proceedings of the International Conference on Data Science and Advanced Analytics*, Oct 2015, pp. 1–7.

[27] W. van der Aalst, A. Bolt Iriondo, and S. van Zelst, *RapidProM : Mine your Processes and not just your Data*. Chapman & Hall/CRC Press, 2018.

[28] F. Bezerra, J. Wainer, and W. M. P. Aalst, "Anomaly Detection Using Process Mining," vol. 29, 01 2009, pp. 149–161.

[29] M. M. Suarez-Alvarez, D.-T. Pham, M. Y. Prostov, and Y. I. Prostov, "Statistical approach to normalization of feature vectors and clustering of mixed datasets," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 468, no. 2145, pp. 2630–2651, 2012.

[30] W. Es-Soufi, E. Yahia, and L. Roucoules, "On the use of Process Mining and Machine Learning to Support Decision Making in Systems Design," in *Proceedings of the 13th IFIP International Conference on Product Lifecycle Management*, vol. AICT-492. Springer, Jul. 2016, pp. 56–66.

[31] A. Ukil, S. Bandyoapdhyay, C. Puri, and A. Pal, "IoT Healthcare Analytics: The Importance of Anomaly Detection," in *Proceedings of the 30th International Conference on Advanced Information Networking and Applications*, March 2016, pp. 994–997.

[32] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Scalable Process Discovery with Guarantees," in *Enterprise, Business-Process and Information Systems Modeling*. Cham: Springer International Publishing, 2015, pp. 85–101.

[33] C. C. Aggarwal, "Outlier Analysis," in *Data mining*. Springer, 2015, pp. 237–263.

[34] M. Hubert and M. Debruyne, "Minimum Covariance Determinant," *WIREs Computational Statistics*, vol. 2, no. 1, pp. 36–43, 2010.

[35] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the Support of a High-Dimensional Distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[36] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*. Association for Computing Machinery, 2000, p. 93–104.

[37] P. J. Rousseeuw and K. V. Driessen, "A Fast Algorithm for the Minimum Covariance Determinant Estimator," *Technometrics*, vol. 41, no. 3, pp. 212–223, 1999.

[38] "Iot security related datasets," https://marketplace.secureiot.eu/marketplace/dataset/, last visited on November 2020.

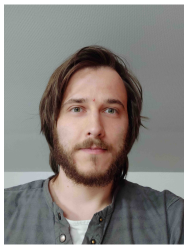
[39] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996, pp. 226–231.

[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.

[41] C. Ferri, J. Hernández-Orallo, and R. Modroiu, "An Experimental Comparison of Performance Measures for Classification," *Pattern Recognition Letters*, vol. 30, no. 1, pp. 27–38, 2009.

[42] Y. Liu, J. Cheng, C. Yan, X. Wu, and F. Chen, "Research on the Matthews Correlation Coefficients Metrics of Personalized Recommendation Algorithm Evaluation," *International Journal of Hybrid Information Technology*, vol. 8, no. 1, pp. 163–172, 2015.

[43] S. Wold, K. Esbensen, and P. Geladi, "Principal Component Analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1, pp. 37–52, 1987, in Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.



Adrien Hemmer received the Master degree in Computer Science from the University of Quebec, Chicoutimi, Canada. At the ENSEM engineering school part of the University of Lorraine, France, he obtained on engineer degree in digital sciences. He is now a second year PhD student in the RESIST research group at Loria-Inria Nancy Grand Est. This research group goal is to make large-scaled networked systems more secure and scalable, leveraging resource elasticity and system resilience. His research interests include the design and evaluation of security solution for Internet-of-Things infrastructures.

Dr. Mohamed Abderrahim received his Ph.D. on computer science and applications from the Institut Mines-Telecom Atlantique, France in December 2018. He also obtained a Master degree from the Ecole Centrale de Lille, France and an engineering degree from the national school of computer science, Tunisia, in September 2015. He worked as a research and development engineer at Orange Labs, France, and currently occupies the same position at INRIA Nancy Grand Est, France in the RESIST research team working on security management.

Dr. Rémi Badonnel is an Associate Professor of Computer Science at TELECOM Nancy, the Engineering School of Computer Science of the University of Lorraine, France, where he is heading the Internet Systems and Security specialization. He is a Permanent Research Staff Member with the RESIST Team at LORIA - INRIA Nancy Grand Est, France. Previously, he worked on change management methods and algorithms for data centers at the IBM T. J. Watson Research Center, USA, and investigated autonomous smart systems at the Oslo Metropolitan University, Norway. His research interests are mainly focused on network and service management, smart and autonomic environments, security and defense techniques, orchestration and chaining of services, cloud infrastructures, and Internet of Things. He has served as TPC co-chair for the IFIP/IEEE IM symposium in 2015, and for the IFIP/IEEE CNSM conference in 2019. He has been elected as chair of the IFIP TC6 WG 6.6 dedicated to the management of networks and distributed systems in 2019.

Dr. François Jérôme obtained a Ms. degree in computer science and received his Ph.D. on robustness and identification of communicating applications from the University of Lorraine, France in December 2009. He was then appointed as research associate at the University of Luxembourg in the SEDAN team of Prof. Thomas Engel. In March 2014, he started as research scientist at Inria in the RESIST team and supports the team leader, Isabelle Chrisment, as deputy leader. His main research area are focused on the use of data analytics techniques for security as well as the definition of SDN-based monitoring probes both at the data and control planes. In 2019, he received the IEEE Young Professional award in Network and Service Management. He is in charge of different international collaborations of the research team with the University of Luxembourg and the University of Waterloo in Canada. In addition to publications, he started as associate Editor-in-Chief of Wiley IJNM and as co-chair of NMRG at IRTF (Internet Research Task Force) in 2019.

Prof. Isabelle Chrisment received the Ph.D. degree in computer science from the University of Nice-Sophia Antipolis, France, in 1996 and the Habilitation degree from Henri Poincare University, Nancy, in 2005. She is a Professor of computer science with TELECOM Nancy Engineering School, University of Lorraine, France. Since 2014, she has been the Scientific Team Leader of the RESIST Team (formerly, MADYNES Team), a joint team between Inria and University of Lorraine. Her main research area is related to network monitoring and security, and especially, within dynamics and large-scale networks.