A Process Mining Approach for Supporting IoT Predictive Security

Adrien Hemmer, Rémi Badonnel and Isabelle Chrisment
Inria Nancy Grand Est - LORIA
Campus Scientifique, 54600 Villers-les-Nancy, France
{adrien.hemmer, remi.badonnel, isabelle.chrisment}@inria.fr

***Abstract*—The growing interest for the Internet-of-Things (IoT) is supported by the large-scale deployment of sensors and connected objects. These ones are integrated with other Internet resources in order to elaborate more complex and value-added systems and applications. While important efforts have been done for their protection, 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 introduce a process mining approach for detecting misbehaviors in such systems. It permits to characterize the behavioral models of IoT-based systems and to detect potential attacks, even in the case of heterogenous protocols and platforms. We then describe and formalize its underlying architecture and components, and detail a proof-of-concept prototype. Finally, we evaluate the performance of this solution through extensive experiments based on real industrial datasets.**

***Index Terms*—Security Management, Internet-of-Things, Process Mining, Data Mining, Anomaly Detection**

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, and industrial infrastructures with the development of industry 4.0. The complexity of systems and infrastructures involving IoT devices is often under-estimated [1], and induces new challenges from a security management perspective [2]. 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) [3]. 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 [4]. More recently in 2019, massive botnet attacks exploited more than 400,000 connected devices against online streaming applications.

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 [2] [4]. The exploitation of one single weak IoT device can be used to take control over a whole network [5]. Traditional security mechanisms, such as intrusion detection systems (IDS), firewalls and antiviruses, are often inadequate with the constrained ressources (CPU, memory, battery) of IoT devices [6]. In addition, these solutions are often specific to given categories of devices and protocols [7], and may fail to address security attacks that occur in complex and heterogeneous environments.

In this paper we propose a process mining approach for supporting the predictive detection of attacks and misbehaviors in IoT-based systems. The solution considers application data generated by IoT devices, in the case of unsupervised datasets, 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 [8]. The approach relies on process mining methods combined with data pre-processing techniques (such as clustering). We describe the architecture and the different components that support this approach. The data pre-processing facilitates the identification of states characterizing the considered IoT-based system. Once these states are defined, the process mining methods permit to elaborate behavioral models of the analyzed system independently from the underlying protocols and device implementations. These behavioral models are then exploited by the proposed approach to detect security attacks in a predictive manner. The solution has been implemented based on a proof-of-concept prototype using the ProM library [9], and is evaluated through an extensive set of experiments.

The main contributions of this paper include: (1) the design of a process mining approach for detecting attacks and misbehaviors in IoT-based systems, (2) the formalization and specification of the different phases supporting this solution, (3) the development of an operational proof-of-concept prototype, and (4) the performance evaluation of our solution based on extensive experiments according to different criteria.

The remainder of the paper is organized as follows. Section II describes existing work related to IoT security and highlight their limitations. Section III describes and formalizes our process mining approach for detecting attacks and misbehaviors in IoT environments. Section IV details our implementation prototype, as well as series of experiments to evaluate the performances of this solution. Section V gives

978-1-7281-4973-8/20/$31.00 © 2020 IEEE

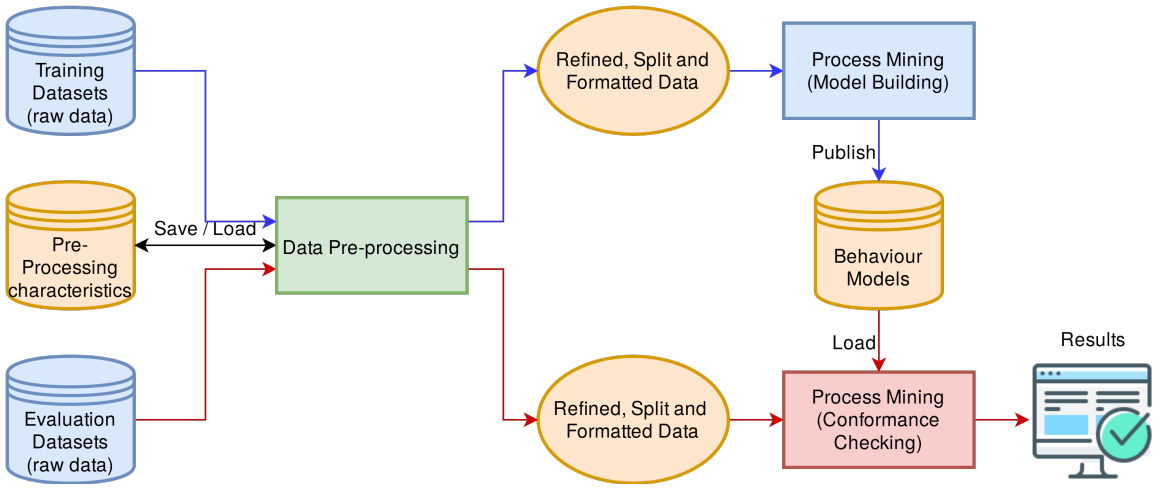

Fig. 1. Architecture of our process mining approach for IoT predictive security

conclusions and points out future research work.

II. Related work

While they have known a growing interest, systems and applications based on IoT devices appear to be more vulnerable to security attacks than those from traditional infrastructures. This phenomenon can be explained by several factors [3]. 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 ressources makes them an attractive target. In that context, they are often affected by naïve weaknesses such as default credentials, poor maintenance and misconfigurations [2] [4]. Moreover, the large-scale distribution of these devices contribute to the presence and multiplication of attack vectors that enable to take the partial or whole control over the considered network [5].

Solutions have already been largely proposed in the litterature to address security issues induced by the Internet-of-things. Amongst them, [10] describes a set of recommanded 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 [11]. 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 [12], 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 [10] 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 [13] and [14] 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 analyzis of application data coming from various sources.

Analytics methods have already been specified to characterize IoT data and infer potential attacks. For instance, the authors of [15] exploit several machine learning methods applied to datasets generated by smart cities. In the same manner, the authors of [16] identify botnet behaviors from a public NetFlow dataset issued from IoT-based systems. First, they standardize the collected data that are then clusterized 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 neural networks, with long short term memory (LSTM) have been experimented in [17]. 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 [18], and are detailed and compared to other data mining techniques in [9]. Typically, they are exploited to detect abnormal sequences of events from specific logs, that may characterize system failures and attack attempts [19].

They may also be combined with machine learning (ML) techniques. In particular, the authors of [20] 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 [21], 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.

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, respectively a model building block and a misbehavior detection block. These two last blocks rely on process mining techniques. 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 raw data have first to be transformed by the data pre-processing block, which is detailed in Figure 2, in order to generate refined data that are interpretable by the model building block. These refined data are then used by process mining algorithms to generate behavioral models. These behavioral models are formally expressed as petri nets representing discrete event models of the observed system. They correspond to bipartite graphs, i.e. their nodes can be split into two disjoint and independent sets. The first set, standing for the places (circles), and corresponding to the states of the system, while the second one, standing for the transitions (boxes), correspond to the events enabling a change of states. These petri nets typically contain one or several token(s) that permit transitions inside the graphs.

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 heterogenous data inputs following

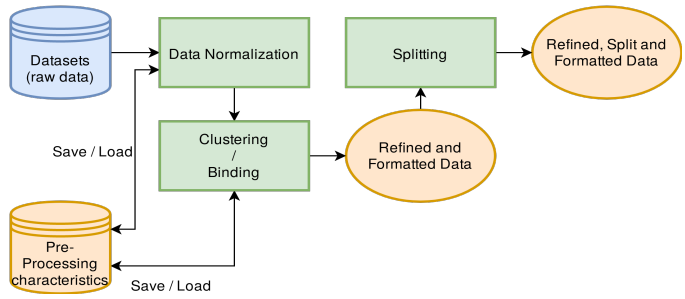

Fig. 2. Detailed view of the data pre-processing block

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, as depicted in Figure 2. 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. In particular, (F_1, F_2, F_3) and (G_1, G_2, G_3) correspond to the same state, if and only if F_1=G_1, F_2=G_2 and F_3=G_3. 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 agregated 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. For instance, a feature ranging from 0 to 100, and another one from 1 to 10 million cannot directly be exploited to calculate distances, as their impact will not be equitable. 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.

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

The normalization parameters are stored into the preprocessing 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 agregated 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.

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

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.

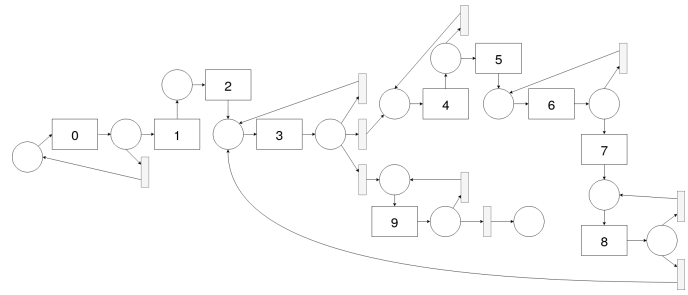

Fig. 3. Extract of petri net obtained by process mining

As a consequence, several tuples may characterize the same state, but at different timestamps. Therefore, when we consider different state identifiers, 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.

$$R_i = \{timestamp : t_i, state : S_p\} \tag{8}$$

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.

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

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 [22], is organized into three main steps. First, it establishes a directly-follow graph from 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. The set of events is then split into two disjoint sets, noted Z_1 and Z_2. Consequently, the refined data are split into two sub-logs, noted L_1 and L_2, where only events from the set Z_1 are mentioned in the sub-log L_1, and only events from the set Z_2 are mentioned in the sub-log L_2. A new cut is performed recursively on the obtained sub-logs, until each event set contains only a single element. Each step causes a cut into the directly-follow graph, the chosen operator corresponding

to the one allowing us to aggregate the highest number of nodes/events. 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_1. An extract of such a petri net is given on Figure 3, where we can observe the different states and transitions resulting from the mining algorithm.

C. Detection of misbehaviors

We will now describe the second phase corresponding to the detection of misbehaviors. The approach consists in analyzing 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 alignement) 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. As an illustrative example, let us assume the behavioral model $M_{example}$ partially described in Figure 3, that we want to compare with the trace T_{example} given by Equation 10.

$$T_{\text{example}} = \begin{cases} \{\{timestamp : t_1\}, \{state : S_1\}\} \\ \{\{timestamp : t_2\}, \{state : S_2\}\} \\ \{\{timestamp : t_3\}, \{state : S_3\}\} \\ \{\{timestamp : t_4\}, \{state : S_9\}\} \end{cases} \tag{10}$$

Table I provides an optimal configuration for this alignment. The + symbol indicates the states in the model and the log that are desynchronized, and for which the alignment cost is incremented by 1. The τ symbol indicates an hidden state of the model that requires to be crossed, so that we can reach the different log states. The optimal cost is therefore of only 1 in this example.

TABLE I
OPTIMAL COST ALIGNMENT

Log Mvmt	+	S_1	S_2	S_3		S_9
Model Mvmt	S_0	S_1	S_2	S_3	τ	S_9

In that context, we consider the fitness metric detailed in Equation 11, 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.

$$Fitness_M(T) = 1 - \frac{Cost(M,T)}{Move(T) + Len(T) \times Move(M)} \tag{11}$$

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. For instance, in our example, $Fitness_{M_{example}}(T_{example}) = \frac{8}{9}$. This metric serves as a support to quantify deviations from behavioral models.

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 12. When a data subset does not fit any behavioral model, then the IoT-based system is considered as misbehaving.

$$\forall i \in [\![1;n]\!], \tag{12}$$

$$Fitness_i = \mathbf{max}_{j \in [\![1;m]\!]} \, Fitness_{M_j}(P_i) \tag{13}$$

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 14.

$$\forall i \in [\![1;n]\!], \exists j \in [\![1;m]\!], \\ RES_i = (P_i, M_j, Fitness_i) \tag{14}$$

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 15 and 16. The presence of one single tuple RES_i is enough to characterize a misbehaving IoT-based system.

$$\forall i \in [\![1; n]\!], \forall r \in [\![1, q]\!],$$
$$NORMAL_r = \{RES_i \mid fitness_i > FT_r\}, \quad (15)$$
$$ABNORMAL_r = \{RES_i \mid fitness_i \leq FT_r\} \quad (16)$$

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.

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 its performances according to different criteria.

A. Experimental setup

In our experiments, we have considered datasets provided by industrial partners, and corresponding to different IoT-based systems, in particular connected vehicles whose results are detailed in the following of the paper. For these environments, we considered two sources of data: the controller area network (CAN) data that include different parameters such as unexpected data frames, and the vehicle-to-everything (V2X) communication data that provide different parameters such as the vehicle speed and its steering angle. A partial extract of some parameters characterising the points of such dataset are given in Figure 4. In our experiments, we considered a dataset of more than 3000 data points (1 MiB) in order to build behavioral models of our IoT-based system. Our evaluation dataset was composed of more than 2500 labelled data points (700 KiB), which serve to quantify the detection performance. The anomalies represents around 20% of the evaluation dataset and have been generated by changing the values of several features in order to simulate the result of some kind of attacks, such as Man-In-The-Middle attack.

The proof-of-concept prototype follows the architecture of our process mining approach for IoT security, and is composed of two main building blocks. 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 performances. 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 whole prototype has been embedded into a docker container. The experiments have been performed over a 3.3 Ghz Intel Core i5 4th generation desktop with 16 GB of RAM. We have evaluated the accuracy of the detection mechanism, as well as the influence of the time splitting and of the pre-processing clustering. We have also compared the solution to other strategies relying on clustering-only and process-mining-only techniques.

```
"fue":          "0.00",
"gear":         "2",
"ignition":     "0",
"rpm":          "1000",
"speed":        "0.00",
"str_ang":      "1.5",
"throttle":     "0.0",
 [...]
"timestamp": "2019-01-16 15:16:51.257437"
```

Fig. 4. Example of parameters characterizing a point from the dataset

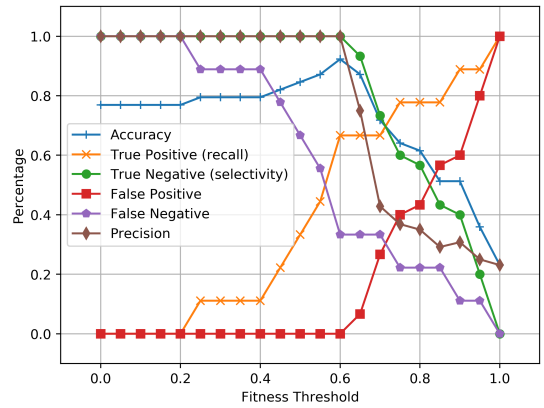

Fig. 5. Detection performances with the following parameters: clustering (k-means, k=8), normalization (min-max), timesplit (30 seconds)

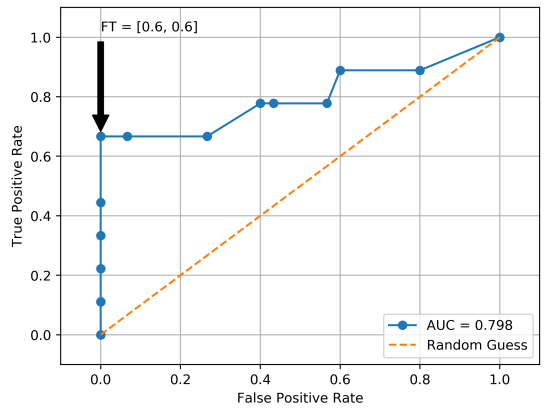

Fig. 6. ROC curve obtained with the following parameters: clustering (k-means, k=8), normalization (min-max), timesplit (30 seconds)

B. Accuracy of the detection

In a first series of experiments, we were interested in evaluating the overall detection performances of our solution. We have considered different clustering (k-means [23], birch [24] and dbscan [25]), normalization (min-max, z-score) and time splitting (from 0 to 300 seconds) techniques. During the experiments, we varied the k parameter of the k-means

clustering technique from 1 to 10, the branching factor of the burch clustering technique from 1 to 1000, and the epsilon parameter of the dbscan clustering technique from 0 to 1. The two considered normalization techniques (min-max, z-score) do not require any specific parameterization. As we expected, the most accurate results have been observed during these experiments, 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. An extract of the behavioral models generated during the building phase was already given in Figure 3. The comparison of the different clustering and normalization techniques in our scenarios has shown that the best detection performances are obtained with the k-means clustering (k=8) coupled with the min-max normalization. These results are detailed in Figure 5 presenting different performance metrics (accuracy, true positive rate, true negative rate, false positive rate, false negative rate, precision), while varying the detection threshold (or fitness threshold) from 0 to 1. We can observe that the true and false positive rates are growing when the detection threshold is increased, while the true and false negative rates are going down. Complementarily, Figure 6 gives the Receiver Operating Characteristic (ROC) curve [26] that characterizes the detection of our IoT security solution. It permits to efficiently determine the most adequate detection threshold corresponding to a value of 0.6, with a minimal false positive rate, in our scenarios. We also provide the corresponding Area Under the Curve (AUC) value estimated to 0.798.

C. Influence of the time splitting

In a second series of experiments we wanted to quantify the influence of the time splitting on the performances of our solution. The time splitting is implemented during the data preprocessing, and is characterized by a time interval at which the dataset is split into smaller subsets. 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, its parameterization may also degrade the overall detection performances. Figures 5 and 6 were presenting the performances with a time split configured to 30 seconds. Further experiments are now detailed in Figure 7, when we vary the time interval from 0 to 300 seconds.

The time interval appears to be an important factor in the processing time needed for both phases, the model building and the evaluation. On one hand, the higher the time interval is, the less models have to be built, and the less it takes time, between 80 and 3 seconds with a time split from 10 to 300 seconds. On the other hand, the performance of the evaluation phase is induced from the number of elements to read and their complexity. In our experiments, the evaluation phase took around 3 minutes for a time splitting of 0.0001 and 300 seconds, whereas it took 40 seconds for a time splitting of 30 seconds.

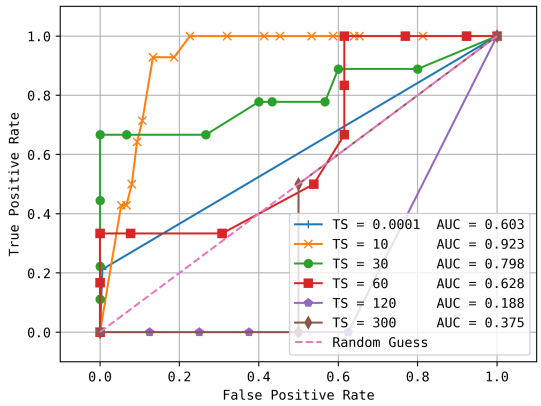

Fig. 7. Influence of the time splitting on the ROC curves

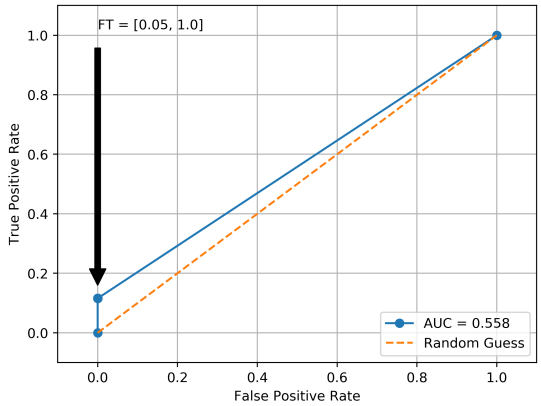

Fig. 8. ROC curve obtained with a clustering-only strategy

When increasing the time interval (from 30 to 300 seconds), we can observe the detection performances are decreasing. When decreasing the time interval (from 30 to 10), the figure shows that the detection performances are increased, but this generates false positive alerts. When the time splitting is too low (with a minimal time interval), there is a high risk to obtain subsets with only one data point, which is not adequate for replaying the traces on the built behavioral models. In such configurations that we should avoid, the process mining does not bring any additional value in comparison to a clustering-only method, such as depicted on Figure 8. In the meantime, considering a too high time interval is also problematic due to the complexity of models, and the characterization of misbehaviors and attacks appear to be less efficient during the detection phase.

D. Influence of the preprocessing clustering

In a third series of experiments we focused on the influence of the preprocessing clustering. We have performed different experiments, while varying the clustering techniques (k-means, birch, dbscan) and the normalization techniques (min-max, z-score). The same techniques (and parameters) are used at the building and detection phases. Changing the parameters of the clustering algorithm has not changed significantly the

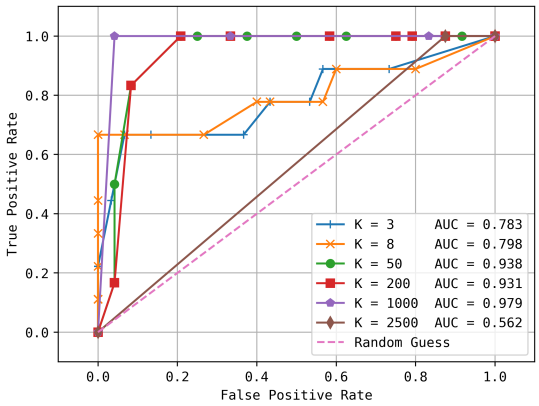

Fig. 9. Influence of the preprocessing clustering on the ROC curves

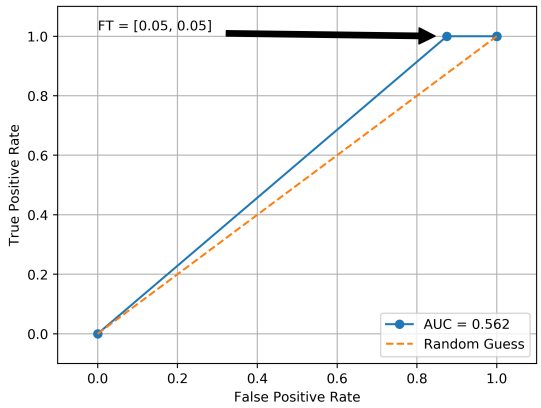

Fig. 10. ROC curve obtained with a process-mining-only strategy

process time. It takes around 30 seconds for each phase except in the limit case (k=2500), where the pre-processing bloc is no longer negligible and takes as much time as the process mining bloc. The most performant detection results have been observed on our dataset with the k-means clustering combined with the min-max normalization. Indeed, amongst our experimental configurations, 86% of the configurations presenting the most efficient ROC curves correspond to the usage of the min-max normalization. In the same manner, 64% of these configurations with the highest results correspond to the k-means clustering.

In that context, Figure 9 synthesizes the different ROC curves corresponding to the k-means clustering combined with the mini-max normalization, while varying the number of clusters (and therefore the number of states) from 3 to 2500 states in order to characterize the IoT-based system. Increasing the k parameter permits to improve the characterization with a large set of states. However, we can also notice that highest k values may degrade the performances. Indeed, the risk is to build too specific behavioral models that are not capable to properly capture deviations. At the extreme case, the highest k values correspond to the lack of data clustering prior to the process mining. This scenario is presented on Figure 10, where the process mining algorithm is directly applied on normalized values. In that case, the detection performances are significantly decreased, as shown by the ROC curve, which is close to the diagonal axis. These different results argue in favor of our IoT security approach which takes benefits from both process mining and clustering techniques.

V. Conclusions

Security management is a major challenge for protecting elaborated IoT-based systems, that are often characterized by heterogeneous protocols and platforms, and that integrate devices with limited ressources (memory, CPU, battery). In this context, we have proposed a process mining approach, that is capable to cope with a large variety of devices and protocols, for supporting IoT predictive security. We have described the underlying architecture and its components, and have formalized the different phases related to this solution, from the building of behavioral models to the detection of misbehaviors and potential attacks. The approach relies on process mining methods combined with data pre-processing techniques, in particular clustering techniques. The data pre-processing permits to identify the states characterizing the IoT-based system, while the process mining methods permit 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 detect misbehaviors and potential attacks in a preventive manner. We have developed a proof-of-concept prototype integrated into a docker container, that implements the proposed solution and exploits the ProM library. We have also performed an extensive set of experiments based on this prototype, in order to evaluate the performances of our approach, with a particular focus on the detection performances, the influence of the time splitting, and the influence of the clustering techniques. We have also compared our approach to two other regular strategies corresponding to clustering-only and process-mining-only techniques. The experimental results clearly show the benefits of our solution combining process mining and clustering techniques.

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 are also planning to consider complementary datasets to experiment our solution, as well as to analyze its potential integration with deep learning techniques. Finally, we would like to investigate incremental model building methods, so that the behavioral models can be enriched at runtime, in order to further improve the performances of such an IoT predictive security solution.

Acknowledgment

This work has been partially supported by the SecureIoT project, funded by the European Unions Horizon 2020 research and innovation programme under grant agreement no. 779899; the exploited datasets have been provided by IDIADA UK.

REFERENCES

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

[2] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the Mirai Botnet," in *Proceedings of the USENIX Security Symposium*, 2017, pp. 1092–1110.

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

[4] 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.

[5] 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.

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

[7] B. Thuraisingham, M. Kantarcioglu, K. Hamlen, L. Khan, T. Finin, A. Joshi, T. Oates, and E. Bertino, "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 (IRI 2016)*, July 2016, pp. 1–10.

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

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

[10] A. Bassi, M. Bauer, M. Fiedler, T. Kramp, R. V. Kranenburg, S. Lange, and S. Meissner, *Enabling Things to Talk: Designing IoT Solutions with the IoT Architectural Reference Model*. Springer Publishing Company, Incorporated, 2013.

[11] M. Pahl and L. Donini, "Securing iot microservices with certificates," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2018)*, April 2018, pp. 1–5.

[12] 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.

[13] 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. [Online]. Available: https://hal.inria.fr/hal-01207859

[14] A. Sehgal, A. Mayzaud, R. Badonnel, I. Chrisment, and J. Schönwälder, "Addressing DODAG Inconsistency Attacks in RPL Networks," in *Proceedings of the Global Information Infrastructure and Networking Symposium (GIIS 2014)*, Sep. 2014, pp. 1–8.

[15] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine Learning for Internet of Things Data Analysis: a Survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161 – 175, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S235286481730247X

[16] 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 (UBMK 2017)*, Oct 2017, pp. 592–597.

[17] 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 (DSAA 2015)*, Oct 2015, pp. 1–7.

[18] 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.

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

[20] 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 (PLM 2016)*, ser. Product Lifecycle Management for Digital Transformation of Industries, R. Harik, L. Rives, A. Bernard, enoit Eynard, and A. Bouras, Eds., vol. AICT-492. Columbia, United States: Springer, Jul. 2016, pp. 56–66, part 1: Knowledge Sharing, Re-use and Preservation. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01403073

[21] 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 (AINA 2016)*, March 2016, pp. 994–997.

[22] 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*, K. Gaaloul, R. Schmidt, S. Nurcan, S. Guerreiro, and Q. Ma, Eds. Cham: Springer International Publishing, 2015, pp. 85–101.

[23] S. Lloyd, "Least Squares Quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, March 1982.

[24] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 1996)*. New York, NY, USA: ACM, 1996, pp. 103–114. [Online]. Available: http://doi.acm.org/10.1145/233269.233324

[25] 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 (KDD 1996)*. AAAI Press, 1996, pp. 226–231. [Online]. Available: http://dl.acm.org/citation.cfm?id=3001460.3001507

[26] M. Zaman and C. Lung, "Evaluation of Machine Learning Techniques for Network Intrusion Detection," in *Proceedings of the IEEE/IFIP International Network Operations and Management Symposium (NOMS 2018)*, April 2018, pp. 1–5.