All Eyes on You: Distributed Multi-Dimensional IoT Microservice Anomaly Detection

Marc-Oliver Pahl
Technical University of Munich
pahl@net.in.tum.de

François-Xavier Aubet
Technical University of Munich
aubet@net.in.tum.de

***Abstract*—The Internet of Things (IoT) is a Distributed System of cooperating Microservices (μSs). IoT services manage devices that monitor and control their environments. The interaction of the IoT with the physical environment creates strong security, privacy, and safety implications. It makes providing adequate security for IoT μSs essential. However, the complexity of IoT services makes detecting anomalous behavior difficult.**

We present a machine-learning based approach for modeling IoT service behavior by only observing inter-service communication. Our algorithm continuously learns μS models on distributed IoT nodes within an IoT site. Combining the learned models within and in-between IoT sites converges our μS models within short time. Sharing the resulting stable models among compute nodes enables good anomaly detection.

As one application, firewalling IoT μSs becomes possible. Combining our autonomous μS modeling with firewalling enables retrofitting security to existing IoT installations. We enable retrofitting access control to existing non-secure IoT installations.

Our proposed approach is resource efficient, matching the requirements of the IoT. To evaluate the quality of our proposed algorithm, we show its behavior for a set of common IoT attacks. We evaluate how domain knowledge enables us to decorrelate events on a node, and how adding context features improves the detection rate.

***Index Terms*—IoT, security, machine learning, modeling, anomaly detection**

I. Introduction

The Internet of Things (IoT) is a Distributed Systems of cooperating Microservices (μSs) [1], [2]. The μSs provide diverse functionality such as acting as gateways towards sensors and actuators, reasoning, or orchestration. By composing different μSs, Pervasive Computing scenarios can be implemented, e.g. for Ambient Assisted Living [3].

Depending on the assumed system architecture those IoT services can run in a distributed way on local IoT sensing and actuation devices, on dedicated edge computing devices, or in the cloud. Our presented solution can run with all topologies.

The IoT is an *attractive attack surface* [4]. It inherently processes security-critical data by collecting information about users or controlling safety-critical machinery. In addition the amount of IoT devices provides an attractive infrastructure for attacking other systems. The Mirai botnet [5] showed the risks that emerge from insufficiently secured IoT μSs. Therefore *securing IoT services is highly required.*

The IoT is a *complex distributed system*: Diverse vendors offer a plethora of devices and services. Vendor comprehensive security is missing.

IoT *installations are long-lasting* with Building Automation Systems easily running for 20 years and more. Keeping the security of the μSs on the devices up to date over their entire lifetime is challenging.

Furthermore, IoT systems *grow and shrink organically*: components are added and removed over time. Adapting security policies with the change is complex. It can often not be done due to lacking on-site experts. In addition the task is complex and therefore error prone even for experts.

As a consequence of the IoT complexity, providing *adequate security to IoT systems is hard* [6]. The typical lack of trained administrator personnel and the inherent complexity *require autonomous solutions* [7]. Already deployed legacy IoT systems often do *not provide state of the art security* [5]. Therefore a brownfield approach with **retrofitting security into existing systems** is required.

A key tool for securing IoT services is providing non-circumventable access control [8]. Such access control requires knowledge about ongoing communication between services.

We present a novel approach for modeling the communication behavior of IoT service purely by observing their traffic. The resulting service-specific communication models can be used for anomaly detection and firewalling. Such firewalling effectively protects IoT services.

We especially address the following **challenges**:

- How to create *detailed* μS *models* by just *observing* their communication traffic?
- How to create *high-confidence* models within *short time*?
- Which *features* contribute how much to the *accuracy* of our communication model?
- How can expected *changes* such as the adding or removal of IoT devices be *anticipated* by the modeling approach?
- How can the solution run on *resource-limited* IoT Things?

For solving the challenges, within an IoT site we install *distributed monitoring and analysis services* on all hosts with sufficient resources. They continuously observe the service communication and update their corresponding μS communication models. Using deep packet inspection and detailed knowledge about the used structured IoT communication protocol enable the creation of feature-rich models.

By using features of the used IoT protocol our *μS models* become *site independent*. This portability enables federating node-comprehensive and even IoT site-comprehensive service behavior analysis results. The models obtained on the

distributed observation nodes can simply be federated. This combination step results in *faster convergence* and a *higher confidence* values of the resulting μS communication models.

Our learned models can be used to identify services that behave unusual, e.g. as they were modified by an attacker. This anomaly detection can be used to inform site operators keeping a human in the loop, or our solution can provide an automated selection of potentially relevant events filtering out the baseline noise of regular IoT communication. The resulting system is on the autonomy level "managed".

The anomaly detection can also be used to configure firewalls to block malicious traffic autonomously. Both forms can be combined, forming an adaptive system where humans set the high-level goals and the system self-manages them.

Our solution operates external to IoT services. It monitors the communication from and to services from the *outside* and can therefore run with standard services, not requiring modification of the deployed (legacy) services. This enables **retrofitting** it into existing solutions resulting in a brownfield approach for *security-by-design* [8].

Our **main contributions** are:

- A portable *human-understandable* per-μS communication model.
- A *scalable* approach for inter-service communication monitoring.
- An *efficient* algorithm for continuously correlating the observations.
- A *feature-dependent accuracy evaluation* of the automated modeling.
- An *open test dataset* the community is missing.
- An *overview on relevant related work* in the currently vital field.

Threat model: Our assumed threat model is that μSs over time change their behavior in an unwanted way. Reasons for a behavior change include *reconfigurations* that could be caused by attacks as well as *updates of the service executables* unfolding unwanted behavior changes. Such an update could be a firmware update on an IoT device for instance. Concrete attacks that we consider are system scans, spying, malicious control, malicious operations, denial of service, data type probing, and wrong setup.

Sec. II introduces our system for creating site-invariant IoT μS models, and our efficient algorithm for periodicity mining. Sec. III describes how our communication models can be used for detecting anomalies in the μS communication. Sec. IV introduces our dataset. It can be reused by others to reproduce our results and compare their performance in the future. Sec. V evaluates our solution quantitatively regarding various aspects.

II. Approach

To illustrate our proposed solution, we developed a prototype. This prototype requires a middleware that enables the mash-up of IoT μSs. With the Virtual State Layer (VSL) a suitable middleware exists [2].

The VSL is the core of the Distributed Smart Space Orchestration System (DS2OS) [9]. The VSL offers service discovery, coupling, and data management. DS2OS additionally offers functionality for service deployment and runtime management.

We use DS2OS as representative base system for illustrating our approach mainly for the reasons:

- DS2OS offers the capability to *distribute cooperating μSs* on multiple IoT nodes,
- its VSL middleware introduces and enforces abstract μS functional interface identifiers (context information). This context enables *correlating findings from different μS instances* within and in-between IoT installations,
- the VSL already *channels all inter-service communication*, providing natural traffic observation points significantly facilitating the inter-μS communication analysis,
- it uses a *REST interface* that is representative for IoT devices, and
- we have *detailed knowledge about the DS2OS system* allowing us to focus on the development of the solution inside a complex real IoT environment taking into account real world needs and feasibility.

The Virtual State Layer (VSL) middleware implements a data-centric discovery and coupling of IoT μSs [10]. Data that is exchanged between VSL μSs always carries a unique sender and receiver identifier. Via the DS2OS service registry this enables *looking up the abstract VSL type identifiers of the services* [11].

The VSL couples services via a REST Application Programming Interface (API) with a *fixed set of commands*: $C_{\text{VSL}} = \{\text{get}, \text{set}, \text{subscribe}, \text{notify}, \text{starttransaction}, \text{commit}, \text{abort}\}$. All *exchanged data nodes* are also *typed* [12], enabling a detailed introspection of the inter-service communication.

The classification of communication partners enables the creation of site-comprehensive *portable communication models*. The fixed API enables *introspection and classification of all* VSL inter-service communication packets. The structured data enables creating *detailed communication models*.

Though basing our solution on DS2OS for the given reasons, the generality of our solution is not limited. In other IoT systems it would though be necessary to introduce additional components that infer the information that DS2OS directly provides. This is possible but introduces additional complexity that we want to avoid for illustrating our approach. As an example for adding such functionality to other middleware, the authors of [13] show a solution for abstract service/ device type classification. Combining such additional analytics with other IoT protocols such as COAP or MQTT provides an equal base for our presented solution. In the evaluation (Sec. V) we show how the absence of certain context information affects the accuracy of our approach. This applies to middleware that does not provide as many features as DS2OS does.

A. System Architecture and Data Collection

We introduce generic monitoring and analysis services that intercept inter-service communication. In the DS2OS system each IoT node that is powerful enough runs an analyzer for analyzing all traffic of the local μSs. These analyzers locally create communication models for their connected services. In

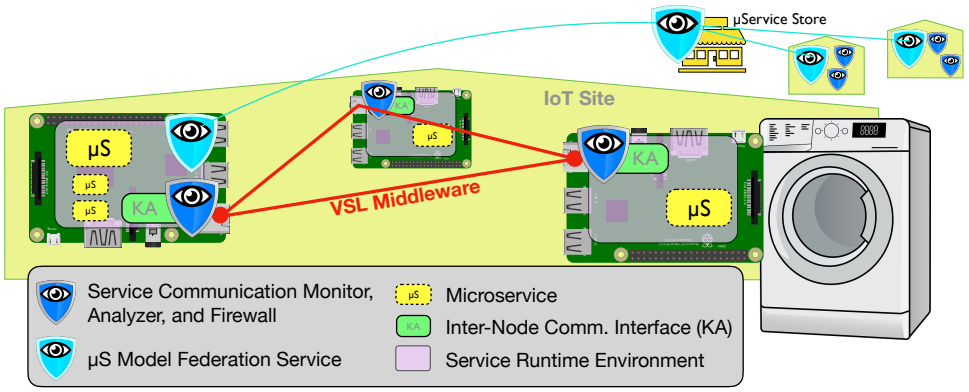

Fig. 1. Our proposed retrofittable security architecture.

our prototype we run them on the VSL middleware access points, the Knowledge Agents (KA) that already manage all inter-service communication [2].

Within each local IoT site we introduce a central coordinator. It is a VSL μS that correlates the findings of the distributed monitors and federates them to site-local service models. The site-local μS models are again correlated to global models in a global IoT μS store. Figure 1 illustrates this.

Combining information from different sites results in meaningful models. High confidence in the observations can be gained within short time. By propagating the resulting enhanced models back, the different connected IoT sites benefit. The quality of their models increases due to a higher amount of observations. Local nodes thereby profit from the distributed observations of their local peers, and of those of nodes in other IoT sites. This is possible as VSL services are typed, enabling the identification of instances of specific services between different sites [12].

The back propagation significantly improves the accuracy of the models. However, the IoT with its heterogeneous links may regularly suffer from connectivity losses that must not decrease the security. Our loosely-coupled modular design ensures that the analyzers on the lowest node level can continue their observation and classification operation even when the upper analysis modules are not reachable.

Once connectivity to the site-local Model Federaration Service (fig. 1) is reestablished, the findings made during the disconnection phase are automatically merged to the site-local μS models, and the global ones. This is possible due to the commutative nature of our combination operation.

B. Feature Engineering

For modeling the μS behavior we select characteristic features. We target creating models that are stable beyond the current μS instance, and that can adapt with the typical changes of IoT environments such as adding or removing IoT devices.

We do not use classic machine learning feature selection algorithms as they do not consider domain specific knowledge but instead find features that are correlated and can be removed without impacting the clustering [14]. As features may change over the lifetime of a μS such functionality does not fit.

Table I lists our selected VSL communication context features, their types, and example values. The table starts with the VSL *instance addresses* of the source and destination μSs that communicate. They are instance-specific and therefore not added to our communication model. The following *types* are the abstract identifiers for offered service functionality in the VSL. The example shows a service providing control to a light switch. The *operation* is one element of the set C_{VSL} (Sec. II). The *type of data* is the type of the accessed data node within the data of the target service [12]. The *value* is the actual value that gets exchanged. The *timestamp* is a value that is roughly synchronized on all participating sites and that monotonically increases over time.

Feature	Type	Example values
Address of source μS	GUID	/../switch943
Address of destination μS	GUID	/../light435
Type of source μS	Categorical	”light switch”
Type of destination μS	Categorical	”light”
Operation	Categorical	”read”
Type of data	Categorical	”number”, ”text”
Value	Type-specific	”6.5”, ”sunny”
Timestamp	Numerical	1520662399723

TABLE I
THE LIST OF FEATURES REPRESENTING EACH ACCESS.

In our evaluation (Sec. V) we show how the availability or absence of a feature influences the accuracy of our model (see Fig. 3).

C. Model Selection for Periodicity Mining

In contrast to regular Internet traffic IoT traffic is more regular [4], [15]. Therefore we include periodicity of communication as relevant part of our model. Similarly to [16], [17], we use inter arrival times of communication packets in our monitor. Having the type of operation as part of our feature vector, we can decompose inter-arrival times down to VSL operation granularity.

Our algorithm uses two types of clustering algorithms, first a grid-based algorithm and then k-means. We use a

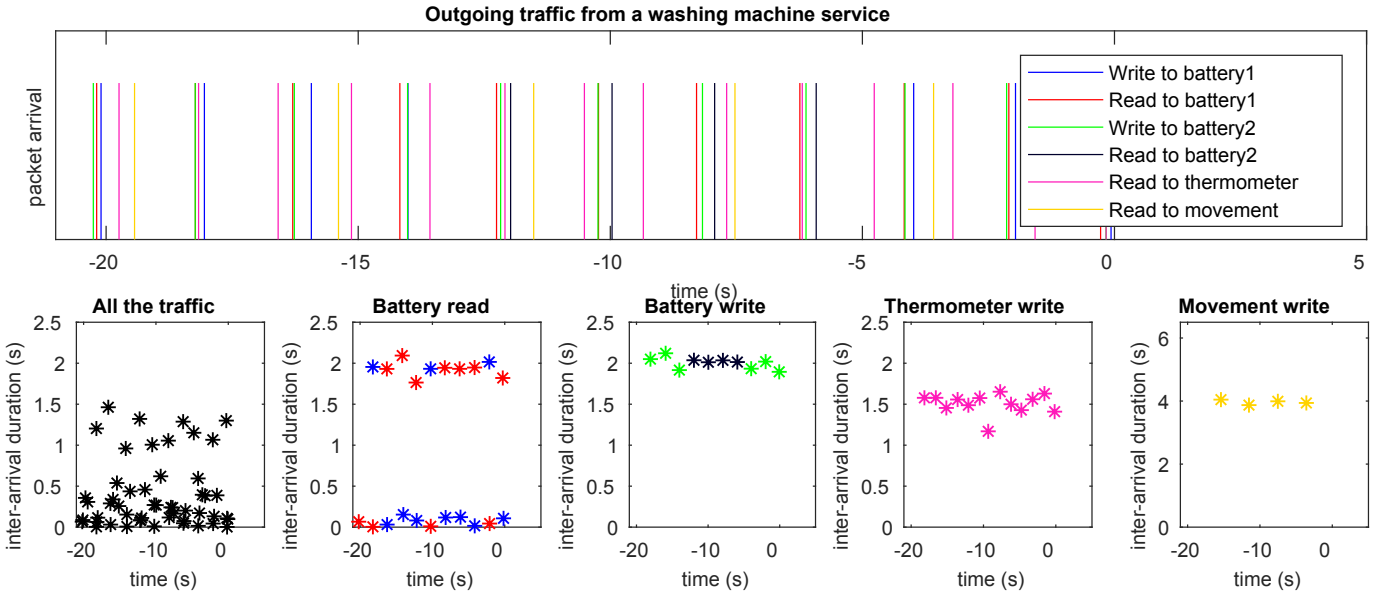

Fig. 2. Decomposition of the outgoing accesses of a node into communication relationships.

sliding window for analyzing the inter-service communication streams. The size of the window determines the resource usage and the time it takes until irregularities are detected.

Storing the last 200 inter-arrival times turned out to be a good compromise between having enough history and being able to react fast. With periodic communication, groups of similar inter arrival times appear. They are clustered based on their density. Fig. 2 illustrates this. The top part shows the observed traffic, the lower part shows the periodicity clusters decomposed by communication partner type, and operation.

We discretize the inter-arrival times into p intervals of similar length. Each interval contains the number of data points that fall into it. The first interval begins at the minimum inter-arrival duration and the last ends at the maximum inter-arrival duration. Clusters are built using intervals that contain at least as many points as the density threshold τ.

The choices of p and τ determine which periodicity can be found. Their determination is the most challenging part of this method [14]. Using our assumption to have periodic traffic, we use a p of 100. As a side effect this simplifies the mapping to a percentage in our model. We use a $\tau = 5$ for a window of 100 data points, corresponding to a cluster containing $\frac{1}{6}$ of the points spread over 3 intervals. We consider such clusters to be the limit of a periodic traffic. These p and τ have empirically proven to be efficient.

If neighboring intervals in the discretized data have a density higher than τ they are identified as a cluster. For each cluster a *cluster center* is placed within the intervals. Once all the clusters are identified, we run one iteration of k-means to center the *cluster center* and get the support of each cluster.

The main advantage of this clustering method is that we can easily find out the number of clusters. If the analyzed communication is not periodic no clusters are found. Otherwise the centers of the clusters correspond to the periods.

Figure 2 shows on top the traffic of one μS. Using the VSL types of the respective communication partners, this traffic can be decorrelated into distinct communication flows. Such decorrelation is straight forward with the VSL as all service communication happens only between a service and its node-local KA interface. The VSL overlay then uses its own data routing mechanisms to serve the data requests [2].

Having decorrelated communication flows, the described periodicity mining reveals the periodicities shown at the bottom of fig. 2. As can be seen, the non-periodic composed traffic on top can be decorrelated into many periodic communication flows. This is typical for the IoT where Things get monitored and controlled on a periodic basis.

D. Modeling: The Communication Model

By monitoring the running service instance's traffic we create a communication model per *site-independent* VSL μS type. It describes the behavior of the μS towards other site-independent communication partner types. The instance addresses from table I are only used to distinguish concrete connections, e.g. when multiple services of the same type run on a node. Table II gives an overview on a μS model.

Partner type (P)	"light" (.6)	"radiator" (1)
# partner type (P)	2.3 (.6)	3.9 (.99)
Same location (P)	yes (.95)	no (.91)
Periods "read"	[25; 2037]	[1020; 1652; 5012]
Periods ...	[...]	[...]
Operation types (P)	"read" (.96)	"read" (.38), "write" (.59)
Data Type	"text" (.98)	"number" (1)

TABLE II
AN EXAMPLE COMMUNICATION MODEL FOR A μS TYPE.

Most entries contain the actual value and its probability. For each operation a set of inter-arrival distances is given. Behind each table entry a cluster of values is stored. This allows us to

know the support of this value and the variance. The clusters can take different forms as detailed in section III.

We use modified BIRCH clusters [18] to save the values. BIRCH clusters are particularly useful in data stream scenarios as they are additive and require low storage for representing large amounts of data points.

Via the additivity, BIRCH clusters match the previously described behavior of enabling autonomous local model creation and refinement *and* global combination of findings at any time. Providing a method inherent compression of measurement value sets, they are optimal for use on resource limited IoT nodes as they require low storage and arew fast to process.

Analog to the proposal of the CluStream algorithm [19] we save the sum of timestamps, and the sum of square of the timestamps additional to the original BIRCH clustering algorithm [18].

The communication models from the analyzers are collected at the Federation Service (fig. 1). Consequently it stores a communication model for each μS type available in the IoT site. Our learning algorithm is *online* (Sec. II-E), updating the communication models continuously.

An advantage of our communication model is that it can run with any subset of the presented features. This makes our solution compatible with IoT middleware that does not provide all context information the VSL does.

Another advantage of our communication model is that it is *meaningful for humans*. In contrast to the machine learning modeling approaches typically used today, the resulting features (parameters) of our approach can be understood. This is relevant for taking humans into the loop.

An example application for the sketched human interaction is asking users if a newly observed and as anomalous classified communication should be allowed. With our features, such a query can become, "Should the climate control service be able to talk to the radiator?" [20]. We expect the accuracy of our approach to increase by adding this expert input step. The human understandable model will be the key of this activity.

E. Online Learning Communication Models

Our approach to learn communication models is inspired by the CluStream algorithm [19] for clustering in data streams. Similarly to [19] we use modified BIRCH clusters with the sum of timestamps and their squares to allow discarding stale clusters. [19]

In our evaluation (Sec. V) the Federation Service (see fig. 1) updates the communication models once a minute. It uses the input communication descriptors from the distributed analyzers. A new BIRCH cluster is created for every feature. The categorical attributes are transformed into vectors using binarization [14]. As the output of the periodicity mining is a list of clusters, we create a list of BIRCH clusters to describe the discovered frequencies. Finally it back propagates the updated model that contains input from all site-local μS instances.

Updating the communication model consists of the operation to merge data from the analyzer with the existing BIRCH cluster in the communication model. With BIRCH clusters this results in simply adding up the clustering features [18].

As proposed in [19], all clusters that have their centers within 3 times the standard deviation are merged. All other clusters remain unchanged. Clusters containing only one point have no standard deviation. In this case we only consider the standard deviation of the other merge candidate.

The behavior of a μS can change over time. Clusters can become stale. To notice such state, timestamps are used. To save storage space, we only store an aggregated value of the timestamps. As proposed in [19] the aggregates still allow to compute the staleness of clusters.

The same procedure is repeated on the top level of our hierarchical communication modeling system The μS store-based Federation Service also obtains the communication models of the connected IoT sites periodically. It combines the contained BIRCH clusters by adding the values and propagates the updated models back to the connected sites. See fig. 1.

III. Execution: Detect Anomalous μS Behavior

The μS type-specific communication models can be used to detect anomalies of μS instances. An anomaly value is computed for each communication message. It is composed of three parts, SoM, SoR, and SoA that are introduced next.

The Support of the Model (SoM) represents how much we can rely on the values saved in the communication model. It depends on the number of contributing communication descriptors ($modelWeight$) and the observation time:

$$SoM = \frac{modelWeight}{maxModelWeight}$$

As the model is updated every minute, the number of descriptors is proportional to the number of μS instances and the monitoring duration. The support is low at the creation of a new model. This represents the fact that in this case we cannot draw meaningful conclusions yet.

We have to limit the growth of the support to enable updates to the model. Such updates reflect when μSs shift their behavior gradually, e.g. when a light sensor that detects sunrise and sunset is used as input for a blind controller and the periodicity of the activities gradually changes with the seasons. At the same time the support has to be big enough to prevent learning anomalous behavior as normal. This problem is known as the stability-plasticity dilemma [21].

We propose a solution using a $maxModelWeight$. This value represents the maximum value of the $modelWeight$. The higher the $maxModelWeight$ the longer it takes to adapt to changed μS behaviors. The lower this value the quicker new behavior becomes part of the μS communication model.

Our hierarchical model update process brings the advantage that even if an attacker manages to take control of all services of one type within a single or few sites, the incoming updates from the other sites still outnumber wrongly learned models on one site. Through the back propagating updates, sites automatically heal their models in that case and will detect the anomalies again.

The *Support of the Relationship* (SoR) represents how relevant a communication relationship is compared to the model. A relationship that was in every descriptor used to build a model is more relevant than another that was observed only a few times. In our current approach we consider infrequent relationships as anomalous:

$$SoR = \frac{modelWeight}{relationshipWeight}$$

The $relationshipWeight$ is the number of descriptors in which this relationship was present. Here again, to allow updates, both values have a maximum of $maxModelWeight$.

The *Support of Access* represents the difference between the current access and previous accesses in a communication relationship. It is computed for each feature of table II:

$$SoA = \frac{dist(center, newValue)}{3 * standardDeviation} * \frac{relationshipWeight}{centerWeight}$$

$center$, $standardDeviation$ and $centerWeight$ are the center, the standard deviation and the weight of the nearest BIRCH cluster. $newValue$ is the value of the current feature for the access operation being evaluated. This value is near or equal to zero if the new access is in line with the normal behavior according to this feature due to the resulting small distance between the two values.

We combine these values to obtain the *Anomaly Value* (AV) as follows:

$$AV = SoM * max(SoR, SoA)$$

We use the maximum here as for an anomalous access either the connection is itself anomalous, or the access is anomalous within the connection. An example for an anomalous connection is a light control μS accessing the door lock. In this case the SoR will be high. An example for an anomalous access within a connection is a light control μS writing the movement value of a movement detection instead of reading it. In this case the SoA will be high. In both cases, the value is weighted with the SoM to express the trust into the regarded value.

High anomaly values obtained by the comparison with a newly created communication model feature may come from normal accesses. For practical reasons, not to block all accesses, we assume that new behavior is normal for newly created communication models. This enables our approach to automatically create models for services that do not have communication models yet, which is especially helpful for the IoT where new services can be expected to emerge frequently.

A. Denial of Service Detection

One of our considered attack scenarios from table III remains undetected by the previous method. Denial of Service (DoS) attacks on allowed relationships that have one periodicity around zero.

In a DoS a high number of accesses happens with high frequency. If for such a communication relationship a periodicity with a very low value has been learned, the single packets of the DoS attack are seen as normal. To detect such group anomalies we compare the list of periodicity clusters on the descriptor with the one of the communication model to obtain an anomaly value:

$$SoA = \sum_{cluster_a \in model} \frac{dist(center_a, center_b) * (weight_a + weight_b)}{3 * standardDeviation_a}$$

Here $cluster_b$ is the nearest cluster to $cluster_a$ in the descriptor frequency list. In the case of a DoS attack a cluster with a very high support will form around 1 or 0. This will cause this modified SoA to be high.

System-inherently, the first packets of a DoS attack will remain undetected. See fig. 4.

IV. Dataset

To the best of our knowledge there is no public standard IoT network traces. Consequently for our evaluation we created our own datasets.

We monitored connections between 7 different VSL service types that connect Light controllers, movement sensors, thermostats, solar batteries, washing machines, door locks, and user smart phones [22]. We captured the traffic of four different emulated IoT sites over 24 hours.

With our first dataset we optimized the parameters of our algorithms such as the duration between two updates of the communication model, or the size of the sliding window. With our second dataset we evaluated the quality of our detection algorithm.

We created two separated datasets to evaluate the sole periodicity mining algorithm containing only one connection. It is generated by two loops in a VSL μS. The two loops perform the same access with different periodicities. We added accesses at anomalous timestamps in the first dataset and three DoS attacks in the second dataset [1]

Table III gives an overview of the attack scenarios that can be found in our provided datasets.

V. Evaluation

In the following we evaluate different properties of our solution and compare the results to state of the art where possible.

A. Performance and Robustness

The hierarchical design makes our solution scale. A node-local detection takes 0.24ms average per VSL message. The independence of the analysis layers and the additive property of the BIRCH clusters make it resilient to temporal link-failures.

The modified BIRCH clustering makes our approach efficient regarding storage and CPU usage. These properties are critical for IoT environments that are typically resource limited. Adding a new point to the dataset of size n and

[1]For enabling the reproducibility of our research we published the three datasets online at www.kaggle.com/francoisxa/ds2ostraffictraces.

Type	Number of packets	Feature used	Description
Network scan	1559	type & number	A μS accesses a range of other μSs.
Spying	532	type & operation	A μS reads values a few other μSs.
Malicious control	889	type & operation	A μS tries to take control over another.
Malicious operation	805	operation type	A μS performs another operation as it should.
Denial of Service	5780	frequency	A μS performs accesses with a very high frequency.
Data types probing	342	data type	A μS writes anomalous data types.
Wrong set up	120	location	A μS accesses a μS in the wrong room.
Total	10027		Corresponds to 3% of the dataset.

TABLE III
ATTACK SCENARIOS IN THE DATASET, INSPIRED BY [23].

dimensionality d requires to iterate though the k existing clusters and then merge it to the corresponding cluster. The complexity of the first operation is the same for k-means and our algorithm: $\mathcal{O}(k)$. However updating the cluster position in k-means requires $\mathcal{O}(n * d)$. Whereas it only requires $\mathcal{O}(d)$ with BIRCH clusters [14].

For d-dimensional data points our modified BIRCH clusters only require $(2{*}d{+}3)$ saved values, independent of the number of points in the cluster. For comparison, using a list for keeping track of all the k points in the cluster would require $k{*}(d{+}1)$ saved values: one value for per dimension plus the timestamp for each data point.

Without using BIRCH clusters and node-local detection, sending the entire traffic log to the coordinator module at each request for getting a detecion result would require on average 169 Bytes per access. Our method only requires to send one descriptor to the coordinator module every minute. The average descriptor size is 96 Bytes. Our descriptor is smaller. More important its size does not increase if the monitored traffic increases.

B. Periodicity Mining Accuracy

In our dataset (section IV), for the detection of single anomalous packets we obtain a detection rate of 93% and false positive rate of 0.3%. With the DoS attacks in the dataset we achieve a detection rate of 95%, an accuracy of 97% and a false positive rate of 1%.

C. Anomaly Detection Accuracy

On the test dataset we achieve a detection accuracy of 99% and a false positive rate of 0.2%. We obtain an overall detection rate of 96.3%. This is caused by the fact that we do not start with communication models but learn them from scratch when a new service appears. Attacks are therefore only detected after this initial phase. When communication models are available and provided by the μS store from the start this factor disappears.

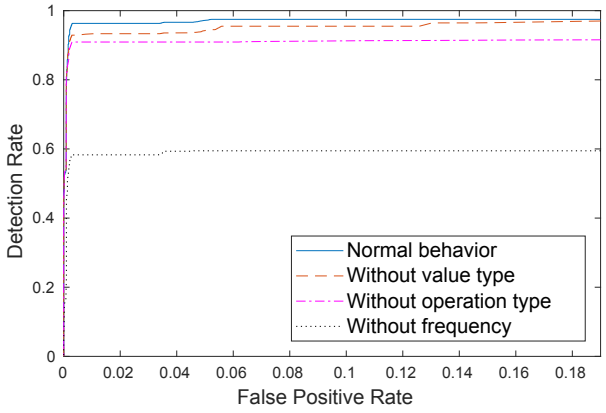

Fig. 3. Comparison of the ROC curves under different circumstances.

Fig. 3 shows the performance of the detection by comparing the detection rate and the false positive rate at different thresholds. The figure also shows the effects of the different features in the communication model on the performance. Even with a reduced feature set the system remains able to detect anomalies at lower accuracy. Figure 3 also shows the positive effect of each feature on the classification accuracy.

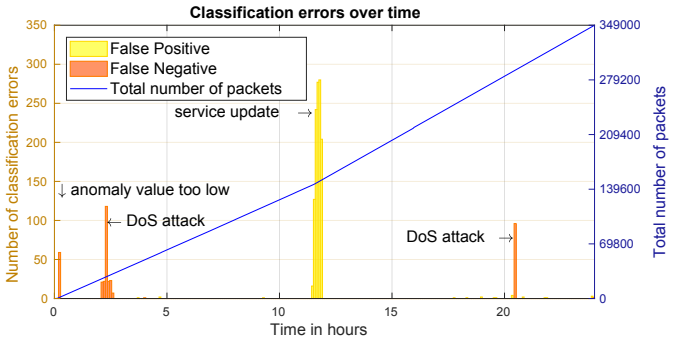

Fig. 4. Classification errors over a day of capture.

Figure 4 shows the classification errors over the day of capture in our dataset. During the capture of the dataset the light controller μSs were updated. Directly after the update, the new accesses are labeled as anomalous. The resulting peak of false positives can be seen at 11 hours. However the model was able to learn this new normal behavior, which allowed a correct classification again. The second and the third small bursts of false negatives correspond to few accesses that are not detected correctly at the beginning of DoS attacks. However, these accesses represent at most 1% of the attack.

In comparison, the authors of [24] obtain an average accuracy of 98%, a detection rate of 96% and a false positive rate of 6%. We obtain better accuracy and false positive rate.

VI. Related Work

Creating models for anomaly detection is a standard technique in the Internet. Recent approaches use machine learning and data mining for implementing Intrusion Detection Systems (IDS) [25]. BotMiner [26] mines similarities in host behavior

within one network to identify participants in a botnet. Though having a different focus, similar to our work the authors perform online clustering of node traffic characteristics for modeling service behavior. The authors of [27] present the benefits of combining models from distributed sites for having significant insights within short time. We employ this principle on the site-local and global Federation Services.

Similar to our Deep Packet Inspection, the authors of [28] analyze REST calls to websites. They try to learn valid communication patterns also using a blackbox analysis approach. Their used methodology is different from our's with symbolic automata in their case and periodicity mining for pattern recognition in our case. We plan to extend our solution with automata to obtain even more detailed models.

Comparing IoT traffic to Internet traffic, the authors of [4] conclude that the IoT has more regular traffic patterns. This motivates the usefulness of the proposed periodicity mining. Our work answers several of the challenges they identified including autonomous adaptation to expected changes in behavior, and building trust by correlating findings from diverse observation points rather than from just one.

For responding to the problem that on-site trained experts are often missing in the IoT, [7] propose to have distributed monitoring points and a central component that does advanced reasoning about the monitored data. In contrast we propose that the monitoring points on the IoT devices perform message analysis on their own. This is more suitable for the IoT as it does not depend on reliable network links, and it enables fast local execution even with high latency IoT links.

Different IDS applications use periodicity mining to detect botnets [16], [17], [29]. Similar to [16], [17], we use inter arrival times. However, these techniques detect periodicity as an anomaly. On the contrary, for the IoT we mine periodicity in order to detect packets that are irregular.

The authors of [30] use operating System Call Frequency Monitoring to model normal behavior and detect anomalies. By counting the amounts of system calls per application run they can detect behavior changes. This is similar to our inter-service traffic monitoring. Our observations are more detailed as we add the dimension time. In contrast to us they do not correlate the analysis results from distributed hosts and sites. They do not adapt to expected changes.

The authors of [31] follow a similar approach to us in another domain, industrial control networks. Like us they employ domain specific protocol knowledge for periodicity mining. They analyze more complex periodicity taking event series in to account. We plan to include event series to our approach as well in the future. Different from us they do not online-update their models, making their solution more static than ours. They also do not correlate findings from distributed monitors.

The authors of [32] use machine learning to fingerprint device types. Their approach can be used to obtain missing context information when applying our solution to IoT systems that do not offer such rich context out of the box like DS2OS.

The authors of [6] also use feature-based clustering in IoT networks to detect anomalous device to device connections. Different from our work their analysis is less detailed as they focus on general network properties instead of protocol specifics. Their models do not contain periodicity and are not human-understandable.

VII. Conclusion

In this paper we introduced our system for creating site-invariant IoT μS models and our novel, highly efficient algorithm for periodicity mining. Our current approach is still limited in modeling complex service behavior. However, the chosen examples with periodic machine to machine traffic are representative for man of today's IoT applications, e.g. automated climate control in smart buildings.

As next step we want to mine more complex periodicity, and we want to integrate an interface that enables improving the accuracy of the communication models via infrequent user interaction.

Another promising direction that emerges from our approach is analyzing the exchanged values as the characterize the concrete control operations. Including them into the communication models could prevent accidental or on-purpose unexpected commands that may result in security or safety risks, e.g. driving a motor for a long time outside its specification.

Our work is an important contribution towards making IoT installations more secure. The fact that many IoT installations are insecure today, and the lack of trained administrator personnel on-site make solutions like ours highly necessary for protecting user security, safety, and privacy. With the long lifetime of IoT installations it is likely that even the few systems that provide adequate security today might be broken tomorrow. Our solution shows a path towards making security-by-design retrofittable into existing IoT installations.

Our solution differs from existing work by operating at μS operation level granularity, and by employing domain knowledge about the used communication protocol. The domain knowledge allows decorrelating observed communication events better. It also allows to create portable μS models that are valid beyond a concrete service instance. Though using machine learning for our analysis, our models remain meaningful for humans to enable keeping users such as site-owners in the loop.

The IoT inherently interacts with our physical ambiance making security risks a much bigger threat than in pure software systems. Examples are Industrial IoT systems controlling heavy machinery, or home IoT services that control security critical systems such as door locks. With our contribution we hope to make the IoT more secure in the future.

References

[1] D. Lu, D. Huang, A. Walenstein, and D. Medhi, "A Secure Microservice Framework for IoT," in *2017 IEEE Symposium on Service-Oriented System Engineering (SOSE*. IEEE, 2017, pp. 9–18.

[2] M.-O. Pahl, G. Carle, and G. Klinker, "Distributed Smart Space Orchestration," in *Network Operations and Management Symposium 2016 (NOMS 2016) - Dissertation Digest*, 2016.

[3] M. Weiser, "The Computer for the 21st Century," *Scientific American*, vol. 265, no. 3, pp. 94–104, Sep. 1991.

[4] N. DeMarinis and R. Fonseca, "Toward Usable Network Traffic Policies for IoT Devices in Consumer Networks." *IoT S&P@CCS*, 2017.

[5] C. Kolias, G. Kambourakis, A. Stavrou, and J. M. Voas, "DDoS in the IoT - Mirai and Other Botnets." *IEEE Computer*, 2017.

[6] I. Hafeez, A. Y. Ding, M. Antikainen, and S. Tarkoma, "Toward Secure Edge Networks - Taming Device-to-Device (D2D) Communication in IoT." *CoRR*, 2017.

[7] N. Feamster, "Outsourcing Home Network Security," in *HomeNets '10: Proceedings of the 2010 ACM SIGCOMM workshop on Home networks*, New Delhi, India, pp. 37–42.

[8] A. Cavoukian, "*Privacy by Design*: Leadership, Methods, and Results." *European Data Protection*, pp. 175–202, 2013.

[9] M.-O. Pahl, G. Carle, and G. Klinker, "Distributed smart space orchestration," in *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*. IEEE, 2016, pp. 979–984.

[10] M.-O. Pahl, "Data-Centric Service-Oriented Management of Things," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, Ottawa, Canada, May 2015, pp. 484–490.

[11] ——, "Distributed Smart Space Orchestration," Ph.D. dissertation, Technische Universität München, München, Jun. 2014.

[12] M.-O. Pahl and G. Carle, "Crowdsourced Context-Modeling as Key to Future Smart Spaces," in *Network Operations and Management Symposium 2014 (NOMS 2014)*, May 2014, pp. 1–8.

[13] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "IoT SENTINEL - Automated Device-Type Identification for Security Enforcement in IoT." *ICDCS*, 2017.

[14] C. C. Aggarwal, *Data mining: the textbook*. Springer, 2015.

[15] R. R. R. Barbosa, R. Sadre, and A. Pras, "Exploiting traffic periodicity in industrial control networks," *International journal of critical infrastructure protection*, vol. 13, pp. 52–62, 2016.

[16] N. Hubballi and D. Goyal, "Flowsummary: Summarizing network flows for communication periodicity detection," in *International Conference on Pattern Recognition and Machine Intelligence*. Springer, 2013, pp. 695–700.

[17] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: detecting botnet command and control servers through large-scale netflow analysis," in *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM, 2012, pp. 129–138.

[18] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," in *ACM Sigmod Record*, vol. 25, no. 2. ACM, 1996, pp. 103–114.

[19] C. C. Aggarwal, S. Y. Philip, J. Han, and J. Wang, "-a framework for clustering evolving data streams," in *Proceedings 2003 VLDB Conference*. Elsevier, 2003, pp. 81–92.

[20] M.-O. Pahl, F.-X. Aubet, and S. Liebald, "Graph-Based IoT Microservice Security," in *Network Operations and Management Symposium (NOMS)*, Apr. 2018.

[21] T. R. Hoens, R. Polikar, and N. V. Chawla, "Learning from streaming data with concept drift and imbalance: an overview," *Progress in Artificial Intelligence*, vol. 1, no. 1, pp. 89–101, 2012.

[22] M.-O. Pahl and L. Donini, "IoT Microservice Security by-Design," in *NOMS 2018*, Apr. 2018.

[23] E. Fernandes, J. Jung, and A. Prakash, "Security Analysis of Emerging Smart Home Applications." *IEEE Symposium on Security and Privacy*, 2016.

[24] I. Hafeez, A. Y. Ding, M. Antikainen, and S. Tarkoma, "Toward secure edge networks: Taming device-to-device (d2d) communication in iot," *arXiv preprint arXiv:1712.05958*, 2017.

[25] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[26] G. Gu, R. Perdisci, J. Zhang, W. Lee *et al.*, "Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection." in *USENIX security symposium*, vol. 5, no. 2, 2008, pp. 139–154.

[27] S. Katti, B. Krishnamurthy, and D. Katabi, "Collaborating Against Common Enemies." *Internet Measurment Conference*, 2005.

[28] G. Argyros, I. Stais, A. Kiayias, and A. D. Keromytis, "Back in Black - Towards Formal, Black Box Analysis of Sanitizers and Filters." *IEEE Symposium on Security and Privacy*, 2016.

[29] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. ACM, 2002, pp. 71–82.

[30] M.-K. Yoon, S. Mohan, J. Choi, M. Christodorescu, and L. Sha, "Learning Execution Contexts from System Call Distribution for Anomaly Detection in Smart Embedded System." *IoTDI*, 2017.

[31] R. R. R. Barbosa, R. Sadre, and A. Pras, "Exploiting traffic periodicity in industrial control networks." *IJCIP*, 2016.

[32] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici, "Profiliot: a machine learning approach for iot device identification based on network traffic analysis," in *Proceedings of the Symposium on Applied Computing*. ACM, 2017, pp. 506–509.