# Graph-Based IoT Microservice Security

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

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

Stefan Liebald
*Technical University of Munich*
liebald@net.in.tum.de

***Abstract*—The Internet of Things (IoT) can be considered as Service Oriented Architecture (SOA) of Microservices ($\mu$S). The $\mu$Ss inherently process data that affects the privacy, safety, and security of its users. IoT service security is a key challenge. Most state of the art providing IoT system security is policy based. We showcase a graph-based access control that runs as module on IoT nodes, or in the network. Our solution intercepts and firewalls inter-service communication. It automatically creates a model of legitimate communication relationships. The model is interactively updated via a simple-to-understand interface. Our solution *adds inevitable IoT security to existing IoT systems*.**

***Index Terms*—IoT, security, passive monitoring, firewall, microservices, autonomous service management, unattended nodes**

## I. Introduction

The Internet of Things (IoT) can be considered as a *Service Oriented Architecture* (SOA) of *microservices* ($\mu$Ss) [1], [2]. IoT devices run software services that communicate with each other in an ad-hoc way to implement IoT applications. An example is a home cinema controller $\mu$S that collects user input and controls the shutters, the lights, the air conditioning, the projector, and the screen.

IoT devices are typically distributed. A significant increase in the computing power of IoT devices is likely to happen [3]. This increase will enable diverse smart devices such as light switches to host multiple $\mu$Ss. IoT systems become distributed computing systems [4].

The IoT inherently processes *privacy*-related data. Its sensors are sensing user presence, and its algorithms are learning user habits. It also processes *safety*-relevant data, e.g. control commands for industrial robots in an industry 4.0 production scenario. Finally the remote control of entities such as smart door locks imposes *security* risks. Consequently, *security is a key challenge for real world IoT systems* [5].

Today, IoT systems are typically secured by configuring access policies and enforcing them. Most current research follows this approach with Policy Decision Points (PDP) and Policy Enforcement Points (PEP) in different locations of an IoT topology [6]. In our other paper at NOMS 2018 [7] we provide a solution for securely distributing access policies in an IoT system of distributed *unattended* nodes.

A problem with policy based approaches is that they typically rely on trusting developers to implement the policies correctly. To mitigate from security holes in policy based approaches, and to enable strong security while assuming untrusted developers, we follow a complementary, self learning in-network approach by

- *monitoring* the communication of IoT $\mu$Ss,
- automatically creating a *communication model* for each $\mu$S, and
- *classifying* inter-service communication traffic as normal or anomalous based on the model.

This research has been supported by the German Federal Ministry of Education and Research (BMBF) in the project DecADe (16KIS0538).



## II. Approach

To implement the previously described steps we introduce traffic monitors. In our demonstrator a monitor runs on each IoT node. However, it could also run on dedicated nodes or active routers and switches in the network. Both is fully independent from the developers of the $\mu$S and thereby enables *introducing security by-design to existing systems* [8].

The monitor intercepts each inter-service communication and does Deep Packet Inspection (DPI). See the shield with the eye on the IoT nodes in Fig. 1. The DPI identifies the communication partners of each data transmission, and additional metadata such as the executed command, the service types, etc. The communication partners are stored as vertices, the communication relationships as edges of our *graph-based communication model*.

Our anomaly detection runs locally on each monitor. Each node maintains its own communication graph. This distributed approach *scales* with the size of an IoT system and provides *low latency*. Both is important for resource limited IoT devices with sometimes time-critical operations.

To bootstrap the communication model of new services, we introduce a *learning phase* similar to the approach of [9] in the Supervisory Control And Data Acquisition (SCADA) domain. In this phase, that happens on the first start of a service, we assume services to behave good and *whitelist all communication traffic*.

A typical IoT site lacks a professional administrator. In addition, the IoT is highly dynamic with frequent topology changes resulting in *high management complexity*. To reduce this complexity our security management is mostly *autonomic*.

To meet the dynamics of IoT systems we introduce an interactive graph updating mechanism. Frequent topology changes, e.g. by adding or removing IoT devices such as a smart washing machine, can especially be expected in the smart home domain where the vendors of technology will try their best to convince consumers to buy new IoT equipment frequently [10].

For the interactive updating of a $\mu$S's communication graph we present site managers with a list containing those communication relationships that were detected but are not part of the model yet. Such a site manager could be a regular inhabitant of an IoT smart space. The presented information is on a high semantic abstraction level to enable non-professional to take meaningful decisions. Approving or denying the prompted communication relationships updates our communication model.

We are intercepting the inter-service communication for our analysis. In this step we also implement a firewall that drops not permitted communication packets. As described before, our self-learning anomaly detection runs independent from security implementations in the $\mu$Ss. It can therefore be combined with multi-vendor IoT systems and complex topologies. The only requirement is knowledge of the used communication protocols for doing the DPI. Together with the active traffic filtering, our solution becomes a *self-learning IoT firewall*. Its *default deny* strategy adds *security by-design to existing IoT systems*.

## III. Demonstrator Description

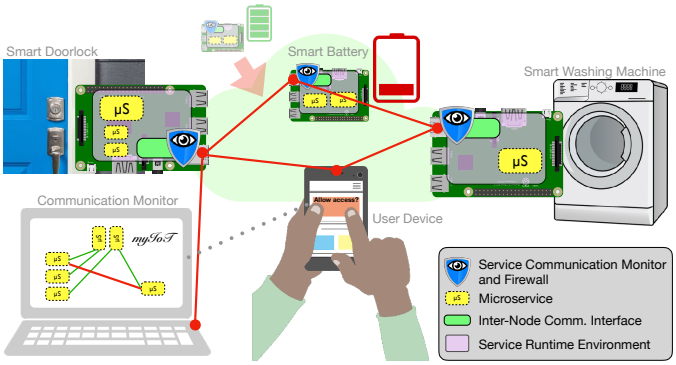


Fig. 1. Demonstrator setup with service monitoring on the bottom left and user interaction in the bottom center.

Fig. 1 shows the demonstrator setup. The IoT site initially consists of three computing nodes. On the left is an embedded controller of a smart door lock, in the middle one of a smart battery, and on the right is one in a smart washing machine.

The IoT controllers are emulated by Raspberry Pis. For service management the Distributed Smart Space Orchestration System (DS2OS) is used [2]. It manages the service placement, the starting, stopping and migrating of $\mu$Ss, etc. For enabling the SOA of $\mu$Ss DS2OS's middleware, the Virtual State Layer (VSL) is used. It provides a service-oriented coupling including the semantic discovery of $\mu$Ss [2].

At the bottom left a laptop is shown. It runs the *dashboard including a visualization of the graph* that shows the currently observed inter-service communication. Edges that are already part of a service's communication model are colored in green. New edges that are not classified yet are shown in orange. Edges that were denied by the site manager are shown in red.

The graph on the monitor shows the live view on the ongoing $\mu$S communication as observed by the distributed monitors. The host-based monitors only know their local context. Besides visualizing the set of automatically created communication models, this component is also capable of analyzing $\mu$S communication on a site-level.

In the bottom center, the *site manager interface* is shown on a smartphone. In our demo each visitor can use her own phone. For communication relationships (edges) that are not yet classified, the interface presents a list of $\mu$S communication endpoints [2]. More concrete the user sees the service names, their semantic, human understandable description, additional metadata, and the violated property of the model such as a non-classified *set data* operation.

In the demo several services are started on the nodes. The control $\mu$S on the washing machine interacts with $\mu$Ss that provide the current state of the local photovoltaic powered battery. Once it is full enough for washing, the machine starts and washes saving environmental resources [11].

In a *first scenario*, by *connecting another smart battery* via another Raspberry Pi, the washing machine control $\mu$S automatically tries to interact with the new battery's $\mu$S. The user is informed and can validate the connection enabling the washing machine to use the new device. This shows how our solution handles *adding an IoT device to a smart space*.

In a *second scenario* the washing machine *control $\mu$S gets updated*. The update introduces malicious behavior that makes the controller access the smart door lock on the left. When our solution is disabled the door suddenly opens.

When repeating the update with our $\mu$S firewall enabled the access is blocked. The site manager gets the corresponding notification and can deny the access. This demonstrates how our solution can *effectively prevent malicious behavior that is introduced by untrusted third party developers*.

The overhead of the described distributed firewall uses resources on each monitor node, and it adds latency to the inter-service communication. The current graph implementation uses about 34 KB for storing the communication relationships of 20 IoT services. The added latency is 6ms per message.

We are currently working on using features for the $\mu$S communication model that are independent of the IoT site, and on reducing the latency of our approach.

## IV. Potential Impact on the Audience

Securing the IoT is essential – especially for preserving user privacy and safety (Sec. I). Security should be part of any IoT implementation from the initial design [8]. Luckily there is research going on for providing IoT systems with such security from the beginning [1], [6], [7].

However, real world deployments show a strong need for *adding security to existing systems*. Malware such as the Mirai Botnet show the malicious potential of unprotected IoT services with free computing resources [12]. In the future even more services that cause even bigger harm will run within IoT spaces [10]. With our demonstrator we showcase adding inevitable and user-friendly security to *existing* IoT systems without a need to make changes in the existing services. Our demonstrator is a major step forward towards a secure IoT.

## 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] R. Want, "When Cell Phones Become Computers," *Pervasive Computing, IEEE*, vol. 8, no. 2, pp. 2–5, 2009.

[4] 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.

[5] S. Misra, M. Maheswaran, and S. Hashmi, "Security Challenges and Approaches in Internet of Things," *Security Challenges and Approaches in Internet of Things*, 2017.

[6] A. Ouaddah, H. Mousannif, A. A. El Kalam, and A. A. Ouahman, "Access control in the Internet of Things - Big challenges and new opportunities." *Computer Networks*, 2017.

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

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

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

[10] M.-O. Pahl and G. Carle, "Taking Smart Space Users into the Development Loop: An Architecture for Community Based Software Development for Smart Spaces," in *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*. New York, NY, USA: ACM, 2013, pp. 793–800.

[11] M.-O. Pahl, H. Niedermayer, H. Kinkelin, and G. Carle, "Enabling Sustainable Smart Neighborhoods," in *3rd IFIP Conference on Sustainable Internet and ICT for Sustainability 2013 (SustainIT 2013)*, Palermo, Italy, 2013.

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