Graph-based Anomaly Detection for IoT Microservices

François-Xavier Aubet, Marc-Oliver Pahl, Stefan Liebald, Mohammad Reza Norouzian
{pahl,liebald,aubet}@net.in.tum.de, norouzian@sec.in.tum.de
Technical University of Munich

ABSTRACT

The Internet of Things (IoT) consists of distributed devices. The devices are managed by microservices that cooperate in an ad-hoc way for implementing diverse use cases. The opportunistic cooperation, and the heterogeneous distributed computing environments make it difficult to manually keep track of the communication relationships between IoT services. We show how a communication graph can be built autonomously, and how it can be used to identify traffic anomalies. A special focus is on bootstrapping the allowed connections of a service. We provide a quantitative evaluation of the added latency of our security feature, and of the graph changes in a real world scenario.

1. INTRODUCTION

The Internet of Things (IoT) consists of distributed computing nodes that serve diverse purposes. For implementing a typical IoT scenario, diverse nodes cooperate. More concrete, microservices running on the nodes work together in a service oriented way [5, 2]. An example is a heating controller service that reads temperatures from distributed meters, and sets the valves of the installed heating devices.

The IoT inherently processes user data, affecting user privacy, safety, and security. Consequently providing security and access control are key requirements for the IoT [3, 6].

Securing the IoT means securing IoT services, or more in particular the data the services exchange [5]. Consequently our focus in this paper is on *securing inter-service communication within an IoT space*. In this paper we focus on the two sub challenges, 1) automatically creating a model for the communication relationships between IoT microservices, and 2) detecting traffic anomalies using the model (Sec. 2).

Most state of the art secures IoT data exchange via policies. Examples are OAuth tokens and group based access control [3, 7, 6]. Our traffic monitoring provides complementary security. It does not require trusting in services correctly implementing the security policies. It is semi-automatic, and requires only limited user interaction (Sec. 3).

PAM 2018 Berlin, Germany
ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

2. APPROACH

We model the microservice communication of each computing node as a graph. Services become vertices, communication relationships edges. The graph is initialized and updated autonomously by node-*locally* observing communication between services. On each IoT node we implement a traffic monitor that intercepts the service communication.

Our initial assumption is that services behave well when first being started [1]. We implement a *learning phase* that updates the local graph with a vertex for each new service, vertices for its communication partners, and edges for all communication relationships. During the learning phase we label these edges as *normal* communication. After the learning phase we classify unknown traffic (edges) as *anomalous*.

IoT topologies are dynamic and therefore the graph has to be updated over time. Site owners get informed of anomalous traffic, e.g. via a message on their smartphones. They can legitimate the unexpected traffic, resulting in a relabeling of the corresponding communication model edge to normal. First tests show that a sufficiently long learning phase results in a low user interaction rate (Sec. 3).

To improve the security we want to remove the initial trust implication and get rid of the learning phase. Our system therefore persists the learned well behavior for each service in its *metadata*. These metadata are exchanged between *all* local IoT computing nodes. When a service was once running inside an IoT space, its communication model can be reused when it is restarted on another node. This *reduces the load* on a node as the learning is not required anymore. The service communication model is valid for all IoT computing nodes within an IoT space. It prevents that updated service binaries suddenly behave malicious.

In our pilot we base our IoT system on the Virtual State Layer (VSL) middleware that implements a tuple space for data-oriented service coupling [5]. Consequently our vertices are VSL service addresses and the communication happens through the VSL entry points, the Knowledge Agents (KAs). We couple our monitor with the KA.

Our communication models use local context such as service addresses. Ideally the communication models would be *portable*, meaning that they can be used for any service instance on any IoT node. The VSL uses a semantic service discovery [4, 5]. Using semantic properties instead of concrete service addresses increases the portability significantly. However, for implementing the described functionality of *automatically* creating *site-independent* communication models we started analyzing more communication parameters including periodicity with machine learning.

3. EVALUATION

Our solution runs on each IoT computing node that hosts services. It is thereby *fully distributed*. Running on each node hosting services, it *scales* with the size of an IoT site. Running the detection *node-locally* matches the *heterogeneous resources* of the IoT. It reduces *latency* and *network overhead*.

The *limited resources* of IoT nodes require a low memory usage. We use stacked hash tables in our analyzer. A vertex requires $100 + number\_edges * 88 * 1.33$ Bytes. The graph data-structure uses $1.33 * size\_vertex * number\_vertex$ Bytes. This results in 33,7 KB for a graph that represents twenty microservices with ten communication relationships each.

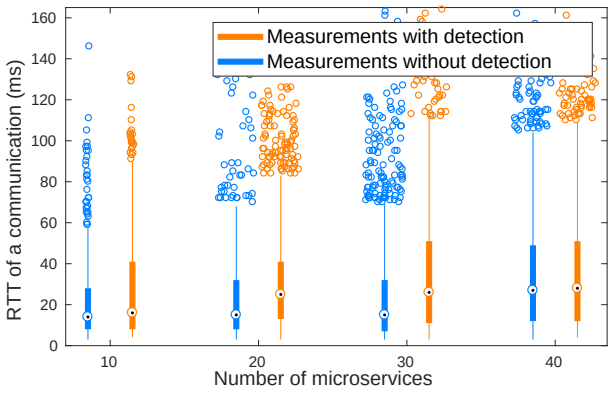

Figure 1: Latency added by our current implementation in inter service communication RTTs.

Fig. 1 shows the latency of the inter-service communication without (left) and with with our solution enabled (right). As can be seen, our Java analyzer delays each VSL communication by about 6ms that are added to the 24ms Round Trip Time (RTT) without anomaly detection. We are currently working on reducing this delay.

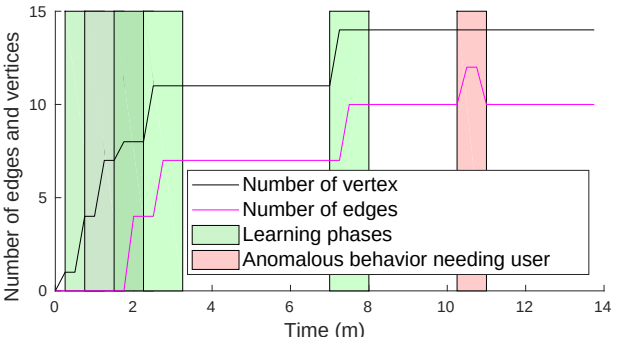

Figure 2: Communication Graph Changes over Time indicating the needed adaptations, and the effectiveness of the learning phase.

Fig. 2 shows the development of a node's connection graph directly after startup where several services are started. During this phase each new service triggers its new learning phase (left side). The green bar starting at minute seven depicts the process of later adding a new service. It triggers a new learning phase for that service.

The red bar after minute ten shows a previously unknown connection. It is classified as anomalous. The user is informed and can give feedback that results in adding the new vertices and edges to the communication model in the service metadata. Such user intervention is seldom since in an IoT space, after a longer running period, the communication relationships typically only change when new entities such as new sensors, are added. To lower the need for user intervention we are currently introducing additional classification features such as periodicity.

4. RELATED WORK

Flow-based Intrusion Detection Systems inspect packets faster as they do not have to do Deep Packet Inspection (DPI). However, especially regarding portability of our communication relation metadata we *require such DPI*.

For industrial Supervisory Control And Data Acquisition (SCADA) systems a similar approach to ours exists [1]. The authors *whitelist flows*, have a *learning phase* but *lack continuous interactive model adaptation*. The IoT requires adaptation as devices are integrated or removed frequently.

We could not identify existing related work that creates portable communication models per service and distributes them among computing nodes.

5. CONCLUSION

We present a graph-based anomaly detection system for distributed IoT microservices. Our approach is self-managing. It adapts to the changes of an IoT site over time by interactively adapting the communication model. Our evaluation already shows that the approach is promising regarding the required low user interaction.

As next steps we want to derive other communication features such as communication periodicity and more complex interaction patterns. Thereby we expect to come up with a better communication model that is more portable as it is independent from local context. We especially target analyzing the IoT service communication over a longer time.

6. REFERENCES

[1] R. R. R. Barbosa, R. Sadre, and A. Pras. Flow whitelisting in scada networks. *Int. j. of critical infrastructure protection*, 6(3):150–158, 2013.

[2] B. Butzin, F. Golatowski, and D. Timmermann. Microservices approach for the internet of things. In *IEEE ETFA*, pages 1–6. IEEE, 2016.

[3] 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.

[4] M.-O. Pahl and G. Carle. Crowdsourced Context-Modeling as Key to Future Smart Spaces. In *IEEE NOMS*, pages 1–8, May 2014.

[5] M.-O. Pahl, G. Carle, and G. Klinker. Distributed smart space orchestration. In *IEEE NOMS*, pages 979–984. IEEE, 2016.

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

[7] S. Sciancalepore, G. Piro, D. Caldarola, G. Boggia, and G. Bianchi. OAuth-IoT: An access control framework for the Internet of Things based on open standards. In *IEEE ISCC*, pages 676–681. IEEE, 2017.