# Access Rules Enhanced by Dynamic IIoT Context

Kevin Wallis, Marc Hüffmeyer, Ayhan Soner Koca and Christoph Reich
*University of Applied Sciences Furtwangen, Furtwangen, Germany*

Keywords: Industrial Internet of Things, Industry 4.0, Context, Rule-based-Security, Context Enhancement Protocol.

Abstract: The Industrial Internet of Things is a fast-growing business with many opportunities but also security risks. For instance, a crucial risk is an attack on customer's or company's internal data to violate the integrity, like order information or machine configurations. In addition, an attacker could inject packets which contain harmful manufacturing machine tool commands. This work introduces a novel rule-based approach which uses subcontexts of the Industrial Internet of Things (IIoT) context for deep packet inspection to protect the privacy of data and to increase the resilience of the system against attacks. Furthermore, formal definitions for the Industrial Internet of Things context, subcontexts and context rules have been investigated and developed. The subcontexts are used in combination with a new protocol called context enhancement protocol, to add context information to packets.

## 1 INTRODUCTION

According to Gartner (2017) in 2020 consists the Internet of Things of more than 20 billion devices, which is a dramatic growth. These interconnected devices enable intelligent information systems, that will be used at different industry sectors like manufacturing, healthcare, entertainment, biotechnology, etc. to offer new services and products. Besides the advantages of such highly interconnected systems, the security challenges (Jing et al., 2014) increase and must be considered. It is a horror scenario of all companies, that after connecting a manufacturing machine to a Cloud infrastructure, an attack can stop the production, corrupt the production of specific orders, read information about company internal machine settings, etc. afterwards. These scenarios lead to a financial loss, image damage or even damage to people.

The paper addresses this problem by introducing a rule-based security approach for gateways which uses context defined attributes and deep inspection to forward valid packets and drop suspicious packets.

The outline of this paper is structured as follows: Section 2 contains related work to context definition and attribute based security especially on Android and Internet of Things. Section 3 defines the terms Industrial Internet of Things, Context, Context Awareness and shows the need for an own context definition for the Industrial Internet of Things. Section 4 gives a formal definition of the IIoT context, the associated subcontexts and suggests a generic approach for gathering different attribute keys. A formal definition of rules and a rule template introduction are given in Section 5. Section 6 describes an infrastructure which supports a distributed access evaluation and could be integrated into existing Industrial Internet of Things environments. Section 7 evaluates the suggested architecture and the Section 8 gives a conclusion and shows an outlook for further research.

## 2 RELATED WORK

Much research about context and context-aware security has been already done in the sector of Mobile-Computing. For instance, Tudoric and Gheorghe Tudoric and Gheorghe (2016) developed a context-aware security framework for Android. This framework validates the current device context towards given XACML (OASIS Standard, 2013) defined policies. Another policy enforcement framework, which uses the context is named CRePE and was published by Conti et al. (2011). In the following three different challenges for these frameworks in the Industrial Internet of Things are explained: $a$) they use a context, which does not fit to IIoT scenarios $b$) they focus on mobile phones whereas IIoT focuses on machines, sensors, gateways, etc. and $c$) they focus on a single device context whereas IIoT merges multiple devices contexts.



Wallis, K., Hüffmeyer, M., Soner Koca, A. and Reich, C.
Access Rules Enhanced by Dynamic IIoT Context.
DOI: 10.5220/0006688502040211
In *Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security (IoTBDS 2018)*, pages 204-211
ISBN: 978-989-758-296-7

Monir (2017) suggested a lightweight attribute-based access control system for IoT which calculates a trust value on the requesting device and only if a minimum threshold value is met, the request is forwarded into the cloud. In addition, the cloud validates the request and the given context with defined policies and only on success the request is forwarded to the sensor. Compared to the approach of Monir, this paper uses user and IIoT specific context, which leads to more specific and secure policies. Furthermore, the acquired context information is evaluated on gateways instead of the cloud.

A security mechanism for IoT which uses context information was introduced by Ramos et al. (2015). The security approach translates raw data into a common data format which could be used for further security steps. This paper contains a new context enhancement protocol which encapsulates other protocols, enriches them with context and renders the above-mentioned translation process obsolete.

Son et al. (2015) introduce a context-based dynamic access control model for ubiquitous sensor networks which uses the intuitive 5W1H (what, where, when, why, who and how) to define a context. The 5W1H approach misses additional information like for example, the order id. For instance, a customer (who) uses the order tracking app (how) of her smartphone (GPS, where) to request progress of her order (why), which is produced from a manufacturing company (what), at 10:00 AM (when) - no order id is part of the defined context.

# 3 THE NEED OF IIoT CONTEXT

The following section defines basic terms like Industrial Internet of Things (IIoT), Context and shows the need of a dedicated context for Industrial Internet of Things.

## 3.1 Industrial IoT

The Internet of Things (IoT) consists of two distinct subsets which were defined by Rodskar et al. (2017) as following: *a*) the consumer IoT which includes wearable computers, smart household devices, and network appliances, and *b*) the IIoT which includes smart systems within processes such as power generation and distribution, manufacturing, medicine and transportation. Furthermore, Rouse (2015) pointed out, that *"IIoT incorporates machine learning and big data technology, harnessing the sensor data, machine-to-machine (M2M) communication and automation technologies that have existed in industrial settings for years"*. This paper focuses on the Industrial Internet of Things.

## 3.2 Context

Winograd (2001) defined context as *"Context is an operational term: Something is context because of the way it is used in interpretation, not due to its inherent properties. The voltage on the power lines is context if there is some action by the user and/or computer whose interpretation is dependent on it, but otherwise is just part of the environment"*. This context definition is used and adapted to the Industrial Internet of Things in Section 4.

## 3.3 Context in Industrial IoT

Based on three different use cases the need of a particular context for the IIoT is shown. Furthermore, important attributes for the context are highlighted.

### 3.3.1 Order Adjustment and Tracking

A company offers individualized product production and allows the customers to track their orders or to do last minute changes to their orders during the manufacturing process. The must be assured that the product tracking and order adjustment is only allowed by the initial **customer**, who gave the **order**. When the order is given a **delivery date** and a **location** need to be set. To adjust or track an order (e.g. change colour before varnishing or check progress), the order needs to be selected by **ID**.

### 3.3.2 Process Outsourcing in Smart Factories

Multiple companies (*A, B, C, etc.*) cooperate and produce together, building a cooperated manufacturing line. In case of a failure in the manufacturing line of company *A*, the failing sub-**process** could be outsourced to a **machine tool** from company *B*, to minimize the delay in production. To outsource a process step the receiving company needs to be identified (e.g. by **id** and **location**) and the **machine tool configuration** should be automatically carried out. Furthermore, the **order** needs to be packed by the packing station with the correct **packaging material** and delivered with a **transport vehicle**.

### 3.3.3 Machine Tool Repairing

A machine tool from the manufacturing line reports a failure **state** and a **technician** has to inspect it and solve the problem. The technician needs remote or

physical access to the faulty machine tool and additional information like the machine **id**, machine **location**, current **material**, current **configuration** and current **tools**.

A sum up of all context attributes can be found in Table 1.

# 4 IIoT CONTEXT

This section describes a novel approach for the definition of the Industrial Internet of Things Context which is the key part of the given rule-based security approach.

## 4.1 IIoT Context Definition

An IIoT context is a union of different subcontexts. Each subcontext $C_I$ is a tuple with an identifier $I$ and a set of key-value pairs $\cup_i(k_i, v_i)$ called attributes. The identifier $I$ is an identification to specify different subcontexts. For instance, a subcontext could correspond to a use case like the request of the progress of order 77 from John Doe or to a user like the customer John Doe. The keys $k_i$ of a subcontext are static and the values $v_i$ are time-dependent which means that they could change their value over time. A subcontext at a specific time $t$ is defined as follows:

$$\begin{aligned} &i,t \in \mathbb{N} \\ &C_I(t) = (I, \bigcup_i (k_i, v_i(t))) \end{aligned} \tag{1}$$

The union of all different subcontexts $C_{I1}, C_{I2}, ..., C_{In} = \cup_j C_{Ij}$ which exist at a specific time $t$ build up the IIoT context $C(t)$.

$$\begin{aligned} &j,t \in \mathbb{N} \\ &C(t) = \bigcup_j C_{Ij}(t) \end{aligned} \tag{2}$$

Due to the time-dependency of the $C(t)$, two different $C(t+1)$ are possible $C(t+1) \neq C(t)$ or $C(t+1) = C(t)$. Furthermore, the number of subcontext over time could also vary depending on the current use cases, users, etc.

At following, an example for the subcontext: progress of order 77 of John Doe ($I = ProgressOrder77JohnDoe = PO77JD$) is given. In general, $t$ is based on the system time and $k_1$ is the user time, so they could vary from each other depending on the time zones, hardware, etc.

- Keys: $k_1 = user\_time, k_2 = user\_location, k_3 = user\_name, k_4 = order\_id, ..., k_n$
- Values: $v_1 = 12:00, v_2 = 48.05, 8.20, v_3 = John\_Doe, v_4 = 7, ..., v_n$
- Attributes (Key-Value Pairs): $(user\_time = 12:00) \cup (user\_location = 48.05, 8.20) \cup ... \cup (k_n, v_n)$

$$\begin{aligned} C_{PO77JD}(t=1) = (&PO77JD, \\ &(usertime = 12:00) \cup \\ &(userlocation = 48.05, 8.20) \cup \\ &... \cup (k_n, v_n)) \end{aligned} \tag{3}$$

## 4.2 Keys of Attributes

Every subcontext is built of a set of attributes (key-value pairs). A list of different attribute keys which were extracted from the use cases 3.3.1 3.3.2 and 3.3.3 are shown in Table 1.

Table 1: Keys of Attributes by Use Case.

| Use Case | Attribute Keys |
|---|---|
| Order Adjustment and Tracking | customer<br>delivery date<br>delivery place<br>order id |
| Process Step Outsourcing in Smart Factories | process<br>machine tool<br>machine id<br>machine location<br>machine configuration<br>order<br>packaging material<br>transport vehicle |
| Machine Tool Repairing | machine state<br>technician<br>machine id<br>machine location<br>current material<br>current tools<br>current configuration |

A generic approach for acquiring attribute keys is needed because the keys from Table 1 are only a subset of all possible attribute keys. To achieve this, groups of different keys are defined and attributes for a subcontext pick the key from them. A key group $K$ is defined as a set of attribute keys, where each key $k$ could only be part of a single group. Furthermore, the union of all attribute key groups contains all possible keys $K_{all}$ from the Industrial Internet of Things context.

$$\begin{aligned} &i,j \in \mathbb{N} \\ &K = \bigcup_i k_i \end{aligned} \tag{4}$$

$$K_{all} = \bigcup_j K_j \tag{5}$$

A lot of attribute keys (e.g. user role, location, etc.) could be taken from the Mobile-Computing context, which was originally defined by Schilit et al. (1994) and enhanced from Chen and Kotz (2000). The following listing shows four different attribute key groups adapted from the enhanced Mobile-Computing context from Chen and Kotz.

- *Computing:* network connectivity, communication cost and bandwidth, nearby resources
- *User:* user profile, user role, location, social situation
- *Physical:* lighting, noise, traffic condition, temperature
- *Time:* time of day, day of the week, timezone, season of the year

Table 1 already pointed out that the Mobile-Computing context is not specific enough and misses keys. For example, depending on the user's role other attribute keys are needed. Additional groups based on the different user's role are listed below. For new user roles the list could be enhanced.

- *Customer:* customer id, orders, etc.
- *Technician:* technician id, machine tools, tools, technical experience, etc.
- *Administrator:* administrator id, network devices, communication between components, etc.

In the Industrial Internet of Things, the production line and the shipping departments are important. The following listing introduces additional attribute key groups containing keys which are inappropriate for the already defined groups.

- *Order:* id, delivery date, place of delivery, manufacturing steps, etc.
- *Machine:* id, location, tools, current order, state, configuration, current material, allocated employee, etc.
- *Employee:* current task, salary, etc.
- *Transport vehicle:* id, vehicle type, status, age, allocated orders, delivery times, driver, etc.
- *Packaging station:* current order, material, employee, etc.

In a nutshell, a subcontext $C_I(t)$ is a tuple containing an identifier $I$ and a union of attributes. Furthermore, each attribute consists of a key $k$ which is element of $K_{all}$ and a time-dependent value $v(t)$.

# 5 RULES AND TEMPLATES

The following section defines formal rules for using the previously defined subcontext $C_I(t)$. Furthermore, rule templates are introduced, that are used as input for generating the active rule set for the gateways.

## 5.1 Formal Rule Definition

This subsection defines rules, which are context dependent. The approach uses the defined attributes from section 4. A rule $R_I$ is always linked to a specific subcontext by the identifier $I$ and the expected values $E_i$ for a specific key $k_i$.

$$\begin{aligned} &i \in \mathbb{N} \\ &R_I = (I, \bigcup_i (k_i, E_i)) \end{aligned} \tag{6}$$

When a subcontext is evaluated, each $v_i$ needs to be an element of $E_i$ from the associated rule.

$$\forall i : v_i \in E_i \tag{7}$$

Examples of rules could be found in the column *Rule* of Table 3.

Rules could be stacked $R_{stacked}$ like an access control list (ACL), so that company compliance rules can be introduced and effect before other rules. The stacked rules are an ordered set, with the company compliance rules at the beginning $R_1, R_2, \ldots$ and the subcontext rules at the end $\ldots, R_{m-1}, R_m$. Each $R_{stacked}$ could be used on a specified set of subcontexts $I_l$. Normally stacked rules are used to enhance a given subcontext rule with company compliance rules, so $|\cup_l I_l| = 1$ and $\left|(R_1, R_2, \ldots, R_{m-1}, R_m)\right| > 1$. Stacked rules can be used to drop requests from none technicians to a machine while the machine is under repair.

$$\begin{aligned} &l, m \in \mathbb{N} \\ &R_{stacked} = (\bigcup_l I_l, (R_1, R_2, \ldots, R_{m-1}, R_m)) \end{aligned} \tag{8}$$

## 5.2 Rule Templates

Rule Templates are used to simplify the generation of rules. They contain the expected attribute keys of a specific rule (see the column *Expected Keys* of Table 2). Equation 9 is the formal definition of a rule template $T$ which is a tuple consisting of an identifier $Z$ and a union of keys $k$.

$$\begin{aligned} &i \in \mathbb{N} \\ &T_Z = (Z, \bigcup_i k_i) \end{aligned} \tag{9}$$

A template can be used to generate multiple rules. The identifier *Z* is not equal *I* otherwise each template could only be used once. For instance, *Z* is the generic *progress of an order* and *I* is the more specified *progress of order 77 of John Doe*. It is possible to reuse *Z* for user Jane Doe but the *I* from John Doe cannot be reused.

# 6 CONTEXT PROPAGATED THROUGH INFRASTRUCTURE

To manage rules (create, revoke, validate) or context (acquire, enhance) a basic infrastructure is needed. This paper introduces a distributed context enhancement and validation approach, which is shown in Figure 1. The infrastructure is split into three layers:

- *Company External* with customers, other factories, other machines, etc.
- *Context Management* with context enhancement, template database, rule database, etc.
- *Company Internal* with production line, machines, sensors, employees, etc.

For example, a request is done by customer, machine or another company. The context enhancement proxy combined with the rule management enhances the incoming request with context. Afterwards, the enhanced request (CEP) is forwarded by the context module (CM). The access is granted if the context is valid at the last component with a CM. This way, it is possible that components between the first and last CM can add additional context. A communication between external and internal does not necessarily run across the context management layer. This paper only focuses on context enhancement at the first component. Now a detailed description of the infrastructure components is given in the following subsections.

## 6.1 Context Enhancement Proxy and Rule Management

Two different services are part of the cloud: context enhancement proxy and rule management. Figure 2 shows the structure of these two components. The context enhancement proxy receives a packet with a context enhancement protocol or a normal request. If the packet is of type CEP the decision maker forwards it to the context module otherwise the request will be processed by template and context acquirer. The template acquirer fetches the associated template from the template database. This template contains the expected attribute keys for the given request, which is

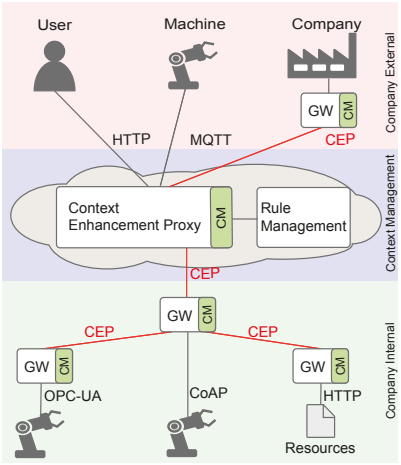


Figure 1: Context Propagation Infrastructure.

further used to acquire the expected context within the context acquirer component. Finally, the context module combines the original request and the acquired context to a context enhancement protocol packet, where necessary.

The rule management contains template and rule database. Generating rules depending on the rule templates or revoking generated rules are necessary operations for the context evaluation. The generation and the revocation are triggered by a security authority. This authority could be an ERP system, a security administrator or a security component. For instance, a process starts and the ERP system triggers the rule generation. The generated rules are stored in the use case rule database with a validity period. For the revocation of a rule, the security authority could request a revocation or the validity end is used.

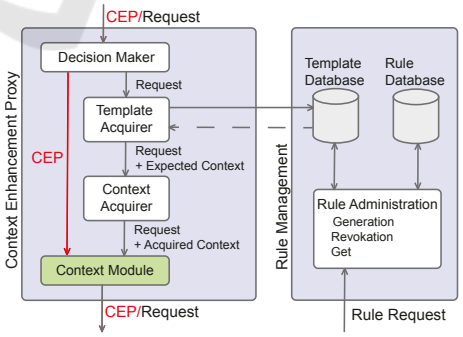


Figure 2: Context Enhancement Proxy and Rule Management.

## 6.2 Template and Rule Database

Templates for rules are used to minimize the error probability for new rules and the needed time to create new rules. The structure of the template database

together with two example entries is shown in Table 2. Three different columns: id, expected keys and template are part of the template database. The identification (id) is needed to reference the correct template while the rule generation. All attributes in the expected attributes column are used for the context enhancement process, each attribute is acquired and added to the context. The template is base for new rules. An entry between square brackets [...] is replaced with the expected values while rule generation, for example *userid* == 77 and entries without square brackets are fixed values like *role* == *customer*.

Table 2: Template Database Example Entries.

| Id | Expected Keys | Template |
|---|---|---|
| Order Progress | User role<br>User id<br>User-location<br>Time<br>Order id | if (<br>role==customer &&<br>userid==[userid] &&<br>userlocation==<br>[userlocation] &&<br>time==[time] &&<br>orderid==[orderid]<br>) then access |
| Machine Status | User role<br>User id<br>User-location<br>Time<br>Machine id<br>Machine status | if (<br>role==technician &&<br>userid==[userid] &&<br>userlocation==<br>[userlocation] &&<br>time==[time] &&<br>m_id==[mid] &&<br>m_status==failure<br>) then access |

Table 3 shows the rule database with two example entries. The id is used to reference the associated rule for the evaluation and the index of the id points out, that a template can be used for more than one rule. Start and end correspond to the validity period, which means order progress$_1$ came into force on 07.07.17 at 12:00 and will expire on 12.07.17 at 12:00. For the rule evaluation, the rule column is used. The left side of the statement is replaced by the context information within the context enhancement protocol, for example *userid* == 7 to 7 == 7.

## 6.3 Context Module

The Context Module (CM) is responsible for the communication between components, which support or do not support context-based security. Figure 3 shows the structure of the CM. After CM receives packets, it checks first in the Next Component Checker whether the next destination of the packet is also a CM. If it is so and the request was already a CEP packet, the packet is forwarded otherwise it will be enriched to

Table 3: Rule Database Example Entries.

| Id | Start | End | Rule |
|---|---|---|---|
| Order Progress$_1$ | 1.7.17 12:00 | 2.7.17 12:00 | if (<br>role==customer<br>&& userid==7 &&<br>userlocation==EU<br>&& time==[12:00-18:00]<br>&& orderid==20<br>) then access |
| Machine Status$_5$ | 3.5.17 07:00 | 5.5.17 17:00 | if (<br>role==technician<br>&& userid==4 &&<br>userlocation==<br>factory area &&<br>time==[07:00-12:00]<br>or<br>[13:00-17:00] &&<br>m_id==15 &&<br>m_status==failure<br>) then access |

a CEP packet and forwarded. The context of the enriched CEP packet contains the acquired context. An acquired context could contain context or none information. If the receiving module is no CM, the context of the request is validated and accordingly forwarded as a normal request or dropped otherwise. For the validation, it is necessary to decapsulate the packet into request and context.

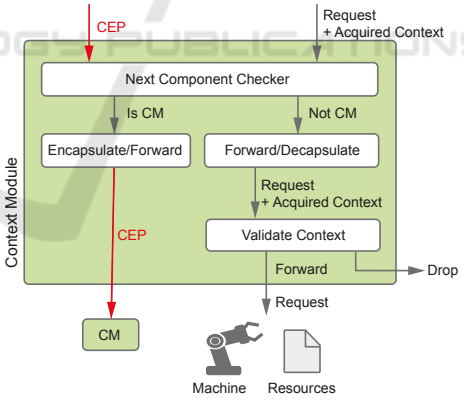


Figure 3: Context Module Structure.

## 6.4 Context Enhancement Protocol

The Context Enhancement Protocol (CEP) is defined in this work to transmit context-enhanced requests between context modules and context enhancement proxy. CEP is an application layer protocol which encapsulates data and adds its own header with protocol/context information. The CEP structure is illustrated in Figure 4. The 4-bit field *TYPE* represents the protocol type of the packet, the field becomes 1 for

request and 2 for a response packet. *VERSION* is the version of the CEP and has a length of 4 bit. The *CONTEXT* part contains an *ATTRIBUTE COUNT* with a length of 1 byte (max. 255 attributes) and the context *ATTRIBUTES* which are using a type-length-value (TLV) approach for their representation. All attribute types have a length of 2 bytes and are picked from the keys of attributes 4.2. The *ATTRIBUTE LENGTH* uses two different length formats: short and long definite from the Basic Encoding Rules (BER) (, ITU-T). If the short form is used bit 8 from the *ATTRIBUTE LENGTH* will be ”0” and bits 7-1 give the length of the *ATTRIBUTE VALUE*. For the long definite form bit 8 has value ”1” and bits 7-1 give the number of additional length bytes which results in a maximum *ATTRIBUTE VALUE* length of $2^{1008} - 1$ byte.

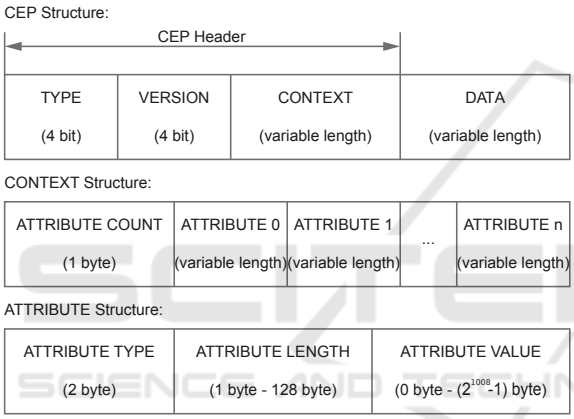


Figure 4: Context Enhancement Protocol.

A central challenge of IIoT is the secure, standardized data and information exchange between devices, machines and services from various sectors. There are numerous communication protocols to choose from to cope with the upcoming tasks on this matter. In the following, four of the most popular protocols for the IIoT are briefly explained.

*a*) *CoAP:* CoAP (Shelby et al., 2014) is a UDP based web transfer protocol specialized for use in resource-constrained devices which provides basic RESTful services and out-of-the-box service and resource discovery. The CoAP protocol is a purely binary protocol and follows the request/response paradigm.

*b*) *HTTP:* HTTP (Fielding et al., 1999) is a generic and stateless protocol which was originally implemented for distributed, collaborative hypermedia information systems.

*c*) *MQTT:* MQTT (Cohn et al., 2014) is a data agnostic, sleek and scalable messaging protocol, tailored for IoT and is based on publish/subscribe paradigm.

*d*) *OPC UA:* The IEC 62541 (IEC, 2008) standard OPC Unified Architecture (OPC UA) is an industrial communication specification that meets all requirements for IIoT communication and is the only recommendation from RAMI 4.0 (Reference Architecture Model for Industrie 4.0) (Hankel and Rexroth, 2015).

## 7 EVALUATION

The effectiveness of the proposed rule-based security approach is presented and evaluated in this chapter using an implemented IIoT scenario order tracking, which is shown in Figure 5. In the given scenario, an automobile manufacturer offers his customers a customized order tracking service in real-time in which she can follow all production steps via cloud-based services in the customer portal. It is not available to customers 24 hours, but only from 12am to 6pm. The cloud-based customer service uses a REST-API (Fielding, 2000) request in the form */customers/7/orders/20/progress*, to get the order progress. The request is encapsulated to CEP and validated against following rule *if (role == customer && userid == 7 && userlocation == EU && time == [12:00-18:00] && orderid == 20)* to prevent malicious behaviour.

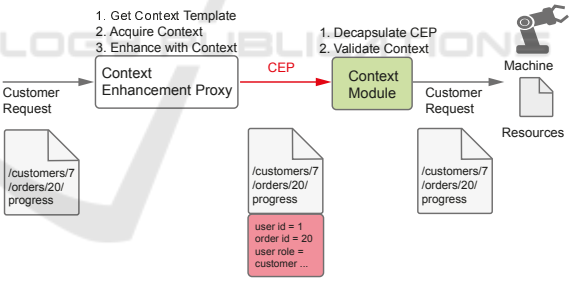


Figure 5: Use Case for the evaluation.

If an attacker logs in as a customer and tries to request information about another customers order, she changes the API call to the following */customers/1/orders/1/progress*. The suggested rule-based security approach the system would deny a response to the attacker because the real user is already logged in as a customer. For the evaluation of the outlined attack scenario, a prototype with python 2.7 and the scapy (Biondi, 2016) library was implemented. The prototype successfully detected the explained attack scenario and dropped the packet.

## 8 CONCLUSION

The paper shows the importance of an IIoT context on basis of three different use cases. Furthermore, it provides formal definitions for the IIoT context, subcontexts and the rules, which are used for the context evaluation. Beside definitions, a generic approach for gathering attributes is shown. This generic approach is split into two generic categories: user and manufactury company attributes. Both use their own independent contexts like user context, order context, machine context, etc. to extract attributes. The paper also suggests a distributed context infrastructure, which uses the defined subcontexts combined with the rules to improve the industrial internet of things security by a rule-based approach. For the approach evaluation, a simplified order tracking scenario was taken and shown, that a malicious behaviour could be prevented.

Further research include the enhancement of the existing infrastructure, so that each context module could add additional context for evaluation. Using content information for the rule templates, rule generation and rule evaluation will be another research topic. In addition to the improved rules, machine learning algorithms will be used to detect suspicious behaviour in the system.

## REFERENCES


Biondi, P. (2016). Scapy. http://www.secdev.org/projects/scapy/.

Chen, G. and Kotz, D. (2000). A Survey of Context-Aware Mobile Computing Research. Technical report, Hanover, NH, USA.

Cohn, R. J., Coppen, R. J., Banks, A., and Rahul Gupta (2014). MQTT Version 3.1.1. *OASIS Standard*.

Conti, M., Nguyen, V. T. N., and Crispo, B. (2011). *CRePE: Context-Related Policy Enforcement for Android*, pages 331–345. Springer Berlin Heidelberg, Berlin, Heidelberg.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis. AAI9980887.

Fielding, R. T., Gettys, J., Mogul, J. C., Nielsen, H. F., Masinter, L., Leach, P. J., and Berners-Lee, T. (1999). Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, RFC Editor. http://www.rfc-editor.org/rfc/rfc2616.txt.

Gartner (2017). Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016. http://www.gartner.com/newsroom/id/3598917.

Hankel, M. and Rexroth, B. (2015). Industrie 4.0: The Reference Architectural Model Industrie 4.0 (RAMI 4.0). *ZVEI: Die Elektroindustrie*.

IEC (2008). IEC 62541: OPC Unified Architecture Specification - Parts 1-100.

(ITU-T), I. T. U. (2015). X.690 - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER). ITU-T 690, ITU-T.

Jing, Q., Vasilakos, A. V., Wan, J., Lu, J., and Qiu, D. (2014). Security of the Internet of Things: perspectives and challenges. *Wireless Networks*, 20(8):2481–2501.

Monir, S. (2017). A Lightweight Attribute-Based Access Control System for IoT. Master's thesis, University of Saskatchewan.

OASIS Standard (2013). eXtensible Access Control Markup Language (XACML) Version 3.0. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html.

Ramos, J. L. H., Bernabe, J. B., and Skarmeta, A. F. (2015). Managing Context Information for Adaptive Security in IoT Environments. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, pages 676–681.

Rodskar, E., Volden, R., and Skjong, E. (2017). Sailing into the Future: Industrial Internet of Things at Sea with X-Connect. *IEEE Electrification Magazine*, 5(3):33–39.

Rouse, M. (2015). Industrial Internet of Things (IIoT). http://internetofthingsagenda.techtarget.com/definition/Industrial-Internet-of-Things-IIoT.

Schilit, B., Adams, N., and Want, R. (1994). Context-Aware Computing Applications. In *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, WMCSA '94, pages 85–90, Washington, DC, USA. IEEE Computer Society.

Shelby, Z., Hartke, K., and Bormann, C. (2014). The Constrained Application Protocol (CoAP). (7252).

Son, J., Kim, J.-D., Na, H.-S., and Baik, D.-K. (2015). CB-DAC: Context-Based Dynamic Access Control Model Using Intuitive 5W1H for Ubiquitous Sensor Network. *International Journal of Distributed Sensor Networks*, 11(9):836546.

Tudoric, C. A. and Gheorghe, L. (2016). Context-Aware Security Framework for Android. In *2016 International Workshop on Secure Internet of Things (SIoT)*, pages 11–19.

Winograd, T. (2001). Architectures for Context. *Hum.-Comput. Interact.*, 16(2):401–419.