DOI: 10.1561/9781680838251.ch7

Chapter 7

# Security of Smart IoT Systems

*By Erkuden Rios, Eider Iturbe, Angel Rego, Saturnino Martinez, Anne Gallon, Christophe Guionneau and Arezki Slimani*

## 7.1 Introduction

Ensuring data confidentiality, integrity, and availability while it is being processed, stored and transmitted by all parts of the environment are high-priority concerns in SIS. One of the most interesting approaches to ensure secure behaviour of the SIS is to embed security features such as monitoring, access control, encryption capabilities, etc. into the IoT Platform used as middleware to capture sensors' data and act as gateway to actuators. As SOFIA-SMOOL, or for short, SMOOL [9], is the IoT platform used in the ENACT Smart Building use case (see Chapter 11), the project has worked in extending this platform with built-in features that enable the platform to implement some of the required security controls to prevent integrity, confidentiality, access control and non-repudiation related issues. Chapter 7.2 describes how the SMOOL platform can be used in the SIS development and operation to monitor and control the desired security properties in the access to resources and communications between smart things of the SIS.

Security assurance at operation does require an external service, agnostic to system design but tailored to final system deployment, that is supervising at all times the security behaviour of the different elements in the IoT system. The role of this security monitoring service is to make sure that security incidents or anomalies are

early identified and corresponding alerts are raised to system operators. Chapter 7.3 describes the ENACT enabler supporting at operations the situational awareness of SIS, the so called, *Security and Privacy Monitoring Enabler*. The enabler is capable of collecting data from different layers of the IoT system: network, system and application layers. All these data are combined by the tool for advanced intrusion detection and anomaly detection. Artificial intelligence detection mechanisms are combined with a multi-layer surveillance so as accurate information of holistic security status of the SIS and all its parts is enabled.

Last but not least, Chapter 7.4 brings an innovative approach to access control in SIS. The tool implementing it is named *Context Aware Access Control* since it offers context-based authentication and authorisation of devices and services exchanging data within the SIS. The chapter describes the various manners in which this tool can be used to secure the IoT accesses, considering contextual information in form of a dynamic risk level computation. The context-awareness capability of the tool has been integrated and validated in the eHealth use case described in Chapter 9.

## 7.2 Built-in Security in IoT Platforms

### 7.2.1 Security-by-Design in IoT Platforms

Complex systems usually cover the security aspects by adding a layer intersecting or covering other business layers (user interface, data management, processes, etc.). When dealing with IoT systems, the heterogeneous nature of sensors, communication channels, Edge devices or Cloud services often demands the architects focusing on business logic, leaving unattended the needed security controls on sensitive areas (e.g. securing the Edge devices, credentials management for key devices,…), even if some sensors may use weak encryption or even produce data in clear because they rely on transport layer encryption.

Therefore, most of the security management is often handled by the developers creating dedicated solutions on the IoT platform. The platform could provide its own battle-tested security mechanisms but those may not fit well or at all with the security features required by the application developers. In these cases, they are impelled to provide additional security measures to the ones available in the IoT platform. And this may bring problems because when custom security is implemented it is likely that flaws occur, particularly when the developer is not a security expert.

We can introduce the security improvements performed in the SMOOL [1, 9] IoT middleware as an example of how adding security features "by design" generates important benefits, like the use of better security patterns, ensuring developer

confidence on the global security, and focus efforts on IoT application logic development and testing, rather than on security aspects.

In this example, we will analyse an application IoT client component exclusively. These components are always the weakest part in potential attacks, since they have limited resources to implement security mechanisms. While servers can also be attack targets, they are usually better prepared and include more or more robust security controls, and changes in the servers are always reviewed and tested exhaustively to prevent scalability problems or vulnerabilities. In SMOOL terminology, a KP (Knowledge Processor) is a client communicating with a SIB (Semantic Information Broker) or server. The KP can send sensor data or actuation orders, and it can also subscribe to messages emitted by other clients. The SMOOL KP clients compose all the messages in Smart Space Access Protocol (SSAP) format (particular of SMOOL ), where the sensors, the data and the metadata are provided in the semantic W3C's Web Ontology Language (OWL) format. This allows other KPs or clients to subscribe to and consume information concepts in the same way for multiple types of sensors. If two different sources such as a complex industrial machine and a simple ambient sensor are providing temperature data to the SMOOL server, another KP could subscribe through the same mechanism to temperature concept in both source KPs to get the temperature value from each.

When embedding security mechanisms in SMOOL IoT platform, three different approaches can be followed, all of them were tested in ENACT and explained below.

#### 7.2.1.1 Custom code of security controls in the KPs

The first implementation of security features within SMOOL clients consisted in adding security metadata to the business data, that is, for example, adding security metadata to the sensing data transmitted by sensors. For instance, if the client was transmitting temperature data (value, unit, timestamp), we could also attach the type and content of security information. These metadata were added as semantic concepts in SMOOL ontology so as they can be published and subscribed to by KPs just the same as KPs do with business semantic concepts. This way, specialized security KPs could only listen to which security data is flowing, instead of subscribing to all sensor types containing security metadata. The new security concepts added in ENACT to the SMOOL ontology covered authentication, authorization, confidentiality, integrity and non-repudiation. They were created to allow flexibility in the exact implementation of the security (for instance, integrity can allow symmetric or asymmetric key-based payloads). The new security concepts are shown in Figure 7.1 just as they appear in the Protégé application [13] used to visualize the ontology.

This way, the KP developers could create sensor KPs that publish sensor data with security information. Other KPs subscribed to the sensor data would check

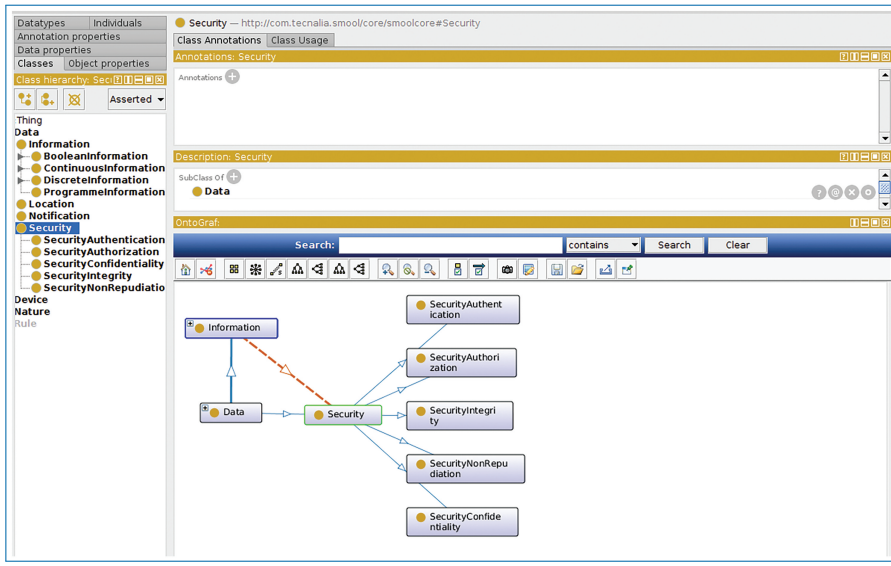


**Figure 7.1.** SMOOL ontology: Security metadata as ontology concepts.

the security data before accepting any values contained in the message. For instance, the publishers of sensor data could add integrity data, while actuation orders would be sent with valid authorization data.

This approach has several potential failures. The first one is that the developer must add extra code to manage security, which means more lines to peer-review and test, and the application implementation would be prone to insecure execution paths when running. The second problem is that the developer could miss some of the things to double-check when using one of the security concepts in the ontology. For instance, the need of salting before hashing encrypted payloads, or the need to authenticate the device before allowing it to publish data. The third issue is that freedom in security coding increases the list of potential security mistakes a non-expert may make and an expert should review.

In Figure 7.2 a simplified version of KP layers is displayed. When a message containing sensor data (in SSAP format) arrives, the first layer is the Comms layer or communications stack, responsible for accepting and assembling the message by using any of the allowed connectors (TCP, Bluetooth, etc.). The second layer is the Model layer where various operations on the message take place, such as parsing, data insertion into the ontology containers, comparison of previous and updated values, etc. These are the core layers, which facilitate the work of IoT application developers but their drawback is that they are black boxes for them. The next layers are the ones created for the real application, including the Custom code layer to collect, send or retrieve sensing data, and the Security layer with the security code. The figure shows these two layers separated logically, although in reality they

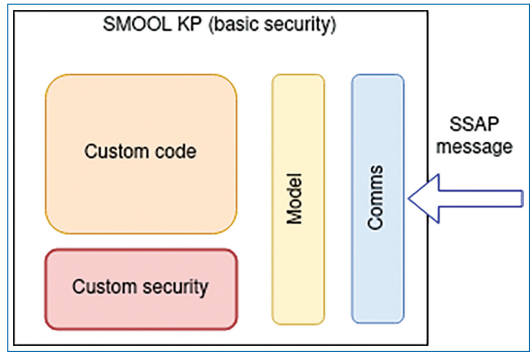


**Figure 7.2.** SMOOL KP (client) layers. Security is handled by the KP developer.

are glued together. The Security layer deals with the management of the security elements. Since these elements are also part of the ontology, in ENACT we have developed a user-friendly API for creating content of these elements.

#### 7.2.1.2 Basic SecurityChecker in the core of the KPs

In the second approach to security for KPs, the aim was to provide a better experience for the KP developer by providing basic built-in security from the KP design, and therefore, preventing potential security flaws introduced by inexperienced developers. Security policy usage philosophy was added to the KP and implemented in the KP core layers. The security control is performed by a SecurityChecker class added in the security layer, which works on all messages received by a KP extracting any security concept present in the message and testing it against a list of policies. If the message does not conform to the policies, the message will be rejected directly from the core layer, so custom code layer will never be aware that the message was received. This solution is more efficient because there is an automatic security check installed on every newly generated KP, since the mechanism comes in the KP design itself. The developer has also fewer lines of security code to implement, because, instead of needing to program the checks of every message for different security constraints fulfillment, some simple one-line rules are defined as policies. For instance, all actuation orders to a specific actuator type (e.g. blinds in a smart building) must contain authorization metadata.

In summary, this second approach, depicted in Figure 7.3, introduces two major differences compared to the previous approach in Figure 7.2. First, the security layer is now part of the core layers, shown as a single vertical layer that works for all the messages, prior to the custom code execution. Second, custom security code is smaller in number of lines, because it would only be dedicated to the enforcement of advanced security features, such as the management of sensitive data, while most of the messages will be filtered or passed by the core security layer.

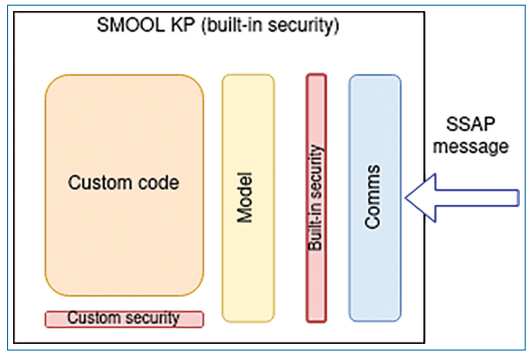


**Figure 7.3.** SMOOL KP (client) layers. Security is enforced for all messages.

#### 7.2.1.3 External SecurityChecker called from the core of the KPs

The third iteration of built-in security in SMOOL IoT KPs takes advantage of the possibility to use enhanced or external security elements to enforce the needed security policies at all times. This way, the security policies would be completely independent from the KPs and adjustable when needed. The approach extends the built-in SecurityChecker of the KPs to provide better security controls. And these controls are still performed in the core layer, in the same manner as in the second approach. When designing the application, these elements are added as a dependency to replace the standard built-in SecurityChecker. The enhanced controls will be loaded when starting the KP.

To demonstrate this, we have used GENESIS for deploying refined security controls from the design phase. Since SMOOL and GENESIS were integrated in ENACT to allow deploying KPs with application extended features, we can also add security features to run either improved extensions of the KP security core layer, or custom security code. Depending on the security needs of the target IoT system, GENESIS will deploy a different implementation of the security policies, but from the design point of view, the declaration of the security enforcement of the policies is the same regardless what checks the external SecurityChecker will do. In Figure 7.4 below, the security core layer is bigger in lines of code. But for the KP developer the security complexity is the same as in the previous iteration.

Now, the core security box is bigger; however, the knowledge about how security is working in our system remains the same, thanks to the use of policies as main concept. Information reaching the custom code can be treated as secure, for all new security upgrades. The security schema remains straightforward, and the application can keep growing by focusing on the business logic features of the KPs (the custom code layer box) rather than security concerns that are handled outside of it.

Therefore, embracing security features of IoT environments from the design phase carries a set of benefits. The most important benefit is that the majority

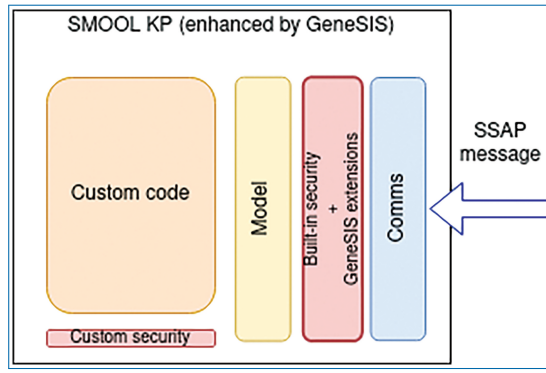


**Figure 7.4.** SMOOL KP (client) layers. Security is enhanced by GeneSIS redeployment.

of the security flaws can be avoided even when building the very first prototype. The software design can provide some security elements as mandatory and block non-secure messages, and all of it in a user-friendly manner. The second benefit is that developers can trust security elements provided from the design phase instead of needing to develop security features based on ad-hoc preferences. The application to be deployed will be more secure and it would be easier to make new releases in the future. Trends in IoT show that security remains as a main concern, and ENACT has demonstrated that IoT applications trusting on security-aware IoT platforms such as SMOOL can be created in a secure manner to prevent most known issues (legacy libraries or software pieces containing vulnerabilities, broken encryption mechanisms, credentials leakage, etc.). This way, securing applications during the design phase can prevent unexpected risky situations when deploying IoT solutions to production environments.

### 7.2.2 Reaction to Cyber Incidents and Anomalies

Smart IoT systems allow devices and data generated to be in the core of the system behaviour, leaving human interaction as trigger elements or passive receptors of actions. In SIS most of the elements must run autonomously, re-adapt based on rules, start and stop things, etc. In fact, the SIS behave as complex ecosystems where elements can have different degrees of intelligence but all of them share a high degree of autonomy. And in these systems, a preventive control monitoring what is happening in terms of security is important, but also a reactive control when things go wrong, i.e. issues are detected. For example, hacking only one of the devices in a SIS could create dangerous situations. Imagine a hacked temperature sensor sending low values to keep a heat system running all the time. Now, imagine a hack of a gas or smoke sensor to forge the sensed dangerous values and prevent them from being detected.

Therefore, apart from monitoring and identifying potential issues and attacks, SIS security must be reactive to take countermeasures in real time. The best way to ensure control is having administration rights on the smart devices and Edge, but not all IoT elements can be controlled (for instance, generic sensors or devices from external vendors ready for plug-and-run). Thus, control must sense the IoT system and must act on it, and in cases where the device cannot be managed from the inside, the control must be done from some other part of the communication or data processing chain.

Some security control systems are ready to detect general problems and react to them. Imagine a new device joins a weak security wireless network. This device could start flooding the communications in the network, creating a denial of service for every other legitimate device. The security control system can detect and block that element, no matter which type of device it is.

Now, a more intelligent and refined malicious IoT device could connect to the same security weak network and send legitimate data shaped in the same format other devices are using in their transmissions. The device is accepted, and the data is also accepted because it fits the format expected to be processed. In this case, a smart control element should understand operational data, so as to be able to detect abnormal values and provide feedback to the Security control system.

In the previous Chapter 7.2, we saw how smart IoT devices could enforce security controls based on policies. The enforcement was done inside the device itself. But not all issues could be detected in the device, and security updates may not be available once deployed. For the security issues not detected and blocked from the IoT devices, we need reactive security mechanisms dealing with them. In ENACT he have developed a reactive security control system that relies on SMOOL platform and its clients as explained below.

Let's go back to the SMOOL IoT platform we described in the previous Chapter 7.2. The clients or KPs connected to the IoT platform exchange complex messages. Some security features are already handled by the core security layer embedded in every KP, and some other security issues are handled by the generic security control system that reacts to incidents notified by the monitoring system described in Chapter 7.3. However, to enforce security in the communications of the SMOOL ecosystem (the server and the smart things connected to it) we have created a Security KP to control what information is really exchanged in the messages of the other KPs, detect non-secure elements, notify the incident, and block the insecure elements.

Figure 7.5 depicts an attack performed by an elaborated rogue application disguised as an IoT client, i.e. a SMOOL KP. The malicious KP behaves as a normal KP so connection to server and message exchange is allowed. Note that other type of rogue KPs will be rejected if their message structure does not conform to the one

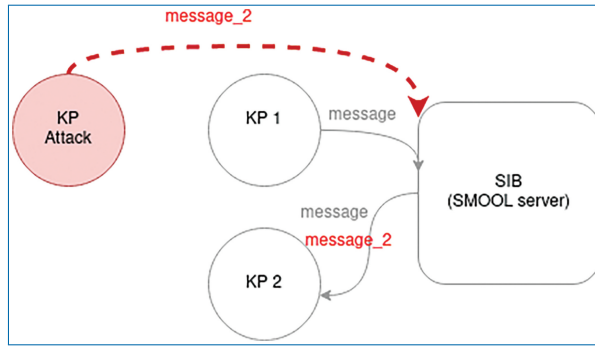


**Figure 7.5.** SMOOL. Malicious application disguised as a KP.

required by the IoT platform. Thus, this attacking KP is a real danger in the system and cannot be detected by the security infrastructure. The client KP2 receives messages from benign KP1 but also malicious messages from KP Attack (in red).

To solve this problem, a Security KP was created which is a special SMOOL client that has the unique ability to access and understand every SMOOL message. The Security KP, being a client rather than a library in the SMOOL server, has another characteristic: it can be upgraded with new features or customized controls faster than if it was allocated inside the server.

Instead of subscribing to all kinds of SMOOL messages and all the ontology concepts, the Security KP can subscribe to a subset of concepts corresponding to those security properties it needs to handle reactions for. Since security metadata was added in the ontology in the same manner as business data, the Security KP can process all or part of the messages to produce faster reactive responses. The first filter could be to check if messages are following the security polices, then inspecting the actual security metadata, and finally, looking for anomalies in the logic or operational data. If any unwanted message is detected, the Security KP has the right credentials to invoke the global Security control system to analyse the metadata or request it to block all communications from the device generating these messages.

Since the IoT server is the real link to the insecure device, a minor implementation for blocking KPs has been developed. This action can be performed only by the global Security control system. Figure 7.6 illustrates how the Security KP can detect refined attacks.

This time, even if the security metadata sent by the malicious KP Attack was fine (and therefore, the KP2 did not reject the message), the other metadata of the message were not compliant with the refined security rule set in the Security KP, so this KP requests the Security control system to block the KP Attack. The Security control system orders SMOOL server to cut off the connection for the offending KP, and add the offending IP to the black-list. The attacker KP would not be able to send further messages because it will not be even able to connect to the server.

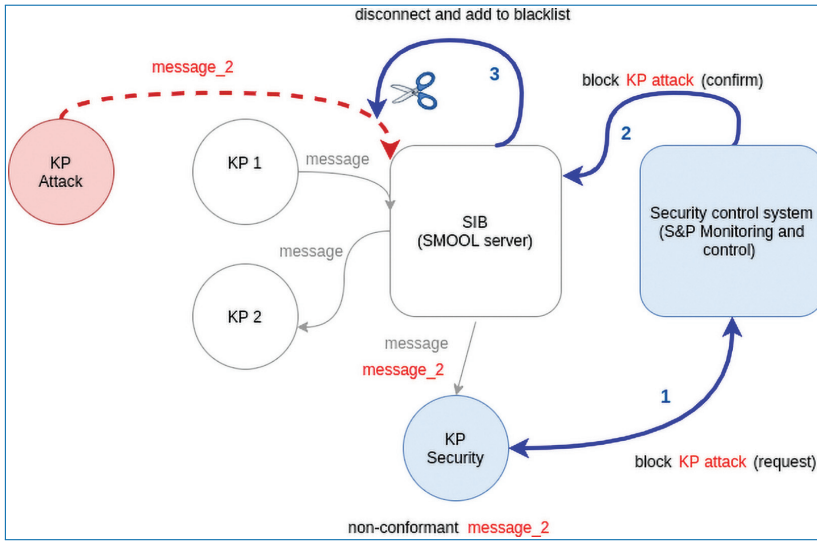


**Figure 7.6.** SMOOL. Malicious KP detected by Security KP.

By using the Security KP, the application-level messages can be tested against a detailed rule set, the IoT data can be transformed into another format that the Security control system could parse, and problems can be detected and blocking orders invoked too. The application can also detect even more sophisticated attacks depending on the use case, because it could have an additional white-list of allowed IoT devices based on historical activity, or detect abnormal behaviour values from legitimate devices, and then request the Security control system to inspect the device.

## 7.3 Continuous Monitoring and Detection in IoT System Operation

Information Security Continuous Monitoring (ISCM) is defined by NIST as *"maintaining ongoing awareness of information security, vulnerabilities, and threats to support organizational risk management decisions"* [7]. In an IoT environment, implementing ISCM provides the security administrator of the SIS with means for continuous situational awareness of the cybersecurity and privacy status of the system. This resource supports the security administrator by identifying cybersecurity incidents along with the targeted assets, as well as informing about the criticality and importance of those incidents so that the security expert can decide on the best cybersecurity strategy to mitigate the cyber threats and protect the assets.

In order to better comprehend the importance of the continuous security monitoring, a review of the NIST Security and Privacy Controls for Information Systems

and Organizations (SP 800-53 Rev. 5) [8] gives the following quick conclusion: at least 68 security controls of the catalogue are explicitly associated with the monitoring activity distributed in 9 control families: *Access Control, Audit and Accountability, Assessment, Authorization, and Monitoring, Configuration Management, Incident Response, Physical and Environmental Protection, Program Management, Risk Assessment, System and Communications Protection, and System and Information Integrity.*

The standard ISO/IEC 27035 identifies multiple technologies as sources of the required security information and events of continuous monitoring as part of detection and reporting phase within the security incident management process [12]. Mentioned technologies include: Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), honeypots, log monitoring systems, security information and event management systems, and network monitoring systems, among others.

Implementing and deploying ISCM mechanisms into IoT environments may become a complicated task due to the high heterogeneity of standards, technologies, protocols and deployment architectures in use. Despite of the complexity, trust models based on security and privacy technologies deployed in IoT systems will be more and more necessary to ensure consumer acceptance [6].

### 7.3.1 Architecture and Main Capabilities

In ENACT, ISCM area is covered by the Security and Privacy Monitoring and Control Enabler (S&P Mon&Con), which aids the SIS operator in learning at all times the security status of the SIS and control the behaviour of the SIS in order to ensure it adheres to the security requirements designed. This Enabler delivers three main capabilities:

- Flexible and extensible continuous monitoring mechanism. A comprehensive security monitoring involves having granular, modular and dynamic security controls to be coordinated with. In this way, the enabler can be adapted to work properly with different types of security controls as data sources, and furthermore, the enabler can even be configured to respond through the use of specific security controls deployed in the SIS itself.
- Advanced anomaly detection through user and entity behaviour analysis using Artificial Intelligence (AI) techniques. Based on a zero trust security model, the enabler follows a cybersecurity strategy of addressing both internal and external threats. Particularly, all internal users and entities are considered for the anomaly-based Intrusion Detection System analytics.
- Scalability of the solution. Both the modular architecture and the technologies the enabler is based on guarantee the solution is able to rapidly scale up in large-scale IoT system scenarios, which also implies the need to deploy

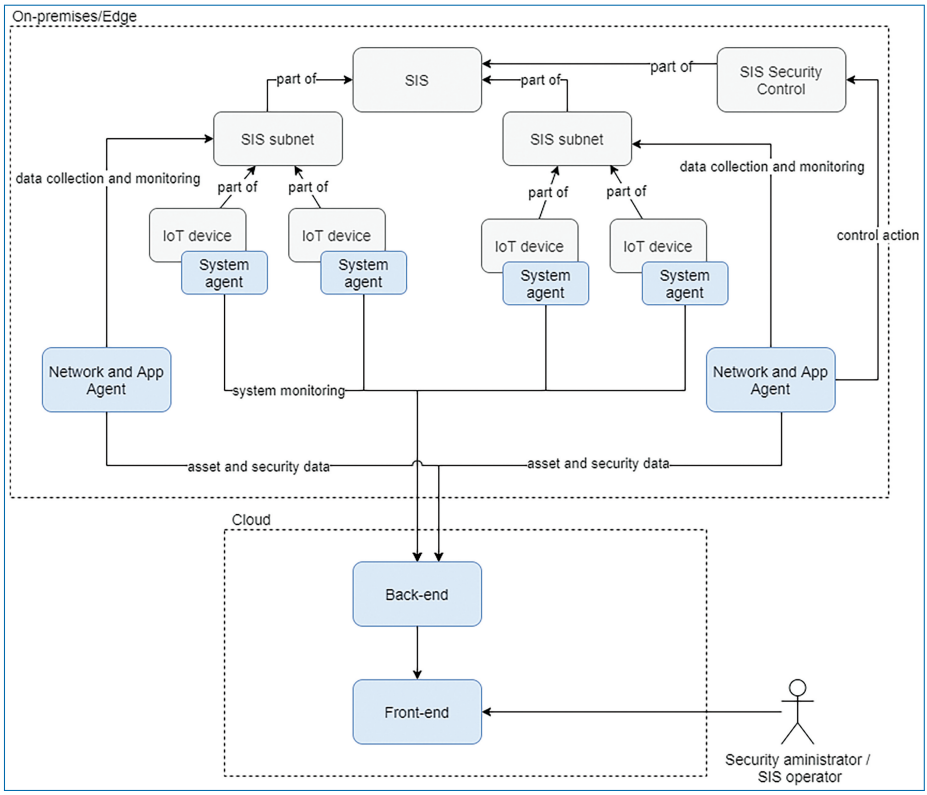


**Figure 7.7.** High-level architecture of the Security and Privacy Monitoring and Control Enabler.

hundreds or thousands of monitoring agents depending on the extension of the system.

Figure 7.7 shows the high-level architecture of the Security and Privacy Monitoring and Control Enabler. The enabler captures different kinds of data from the SIS through multiple distributed probes named *agents*. There are three types of monitoring agents:

1. *Network agent*, which captures network traffic data, network related security events and asset related data. It integrates an open source signature-based network IDS that is able to detect well-known security attacks and generate security events accordingly. Besides that, the agent includes capabilities to generate protocol-specific security events, e.g. related to ARP protocol so as to enable the detection of ARP spoofing attacks.
2. *System agent*, which gathers log information and data related to the activities and processes within the devices of the SIS; and,

3. *App agent*, which collects data at application layer and generates security events accordingly. This agent can be customized depending on the SIS characteristics; e.g. if the SIS is in intensive use of the MQTT protocol, this agent can be developed with specific MQTT based rules for monitoring and controlling that only authorized users and assets (smart things, devices, services, etc.) can communicate in the SIS.

All the data gathered by the monitoring agents is sent to the back-end of the enabler. Depending on the size of the SIS, the amount of data recovered after some time can be huge which would likely cause processing difficulties. In order to avoid a bottleneck at next phases of data processing and analytics, the entry of the back-end is implemented by a streaming bus (based on the open source distributed streaming platform Apache Kafka [4]. Moreover, the streaming bus provides extensibility to the solution by offering multiple channels where various kinds of data can be collected, and it also allows exchanging the outcome from the enabler in form of identified security events with external components.

The back-end of the enabler includes a data storage infrastructure which stores in a NoSQL database (based on Elasticsearch [2]) all the acquired and pre-processed data from the agents. Multiple indexes are created in the storage infrastructure depending on the different sources of data. For example, in a Smart Building IoT system where many communication protocols can be working at the same time, the network data can easily be stored with the definition of approximately 1800 network attributes as shown in Figure 7.8. Bearing in mind that network traffic is only one of the multiple data sources considered for the analytics within the enabler, dealing with enormous amounts of heterogeneous data is one of the major challenges addressed by this enabler in IoT environments.

The main ground-breaking part of the enabler is the anomaly detection service included in the back-end. AI techniques have been leveraged to analyse the collected SIS data and security events in order to detect anomalies in the system. Mainly, unsupervised Deep Learning techniques have been used to perform the SIS behavioural analytics and anomaly detection. Many Deep Learning techniques have been studied depending on the SIS characteristics so as to implement an accurate anomaly-based detection system. Figure 7.9 depicts Vanilla Long short-term memory (LSTM), a type of recurrent neural network (RNN) architecture, predictions for MQTT protocol in a Smart Building system showing as red points the anomalies detected correctly, where real MQTT traffic behaviour deviates largely from predicted one.

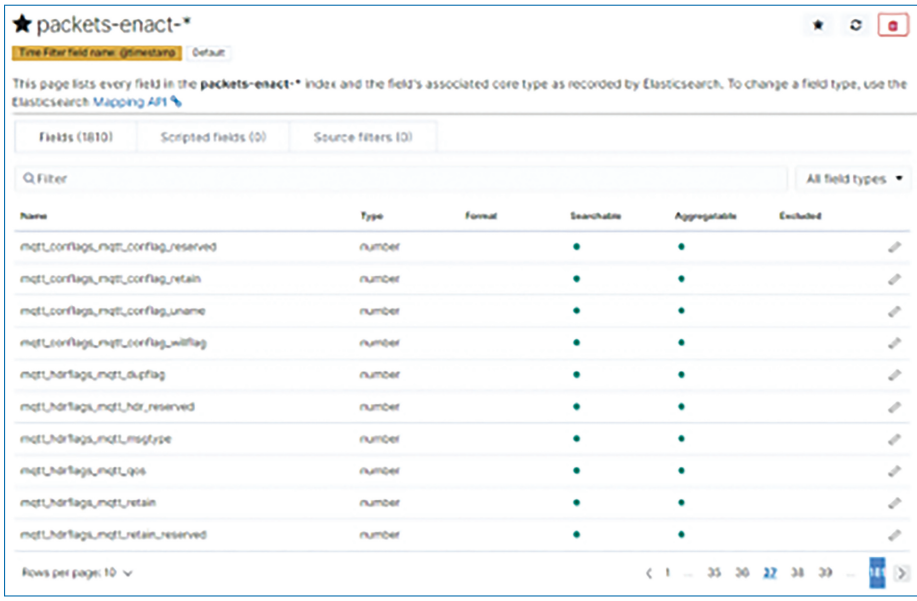


**Figure 7.8.** Extract of the network fields of stored indexes in the Smart Building.

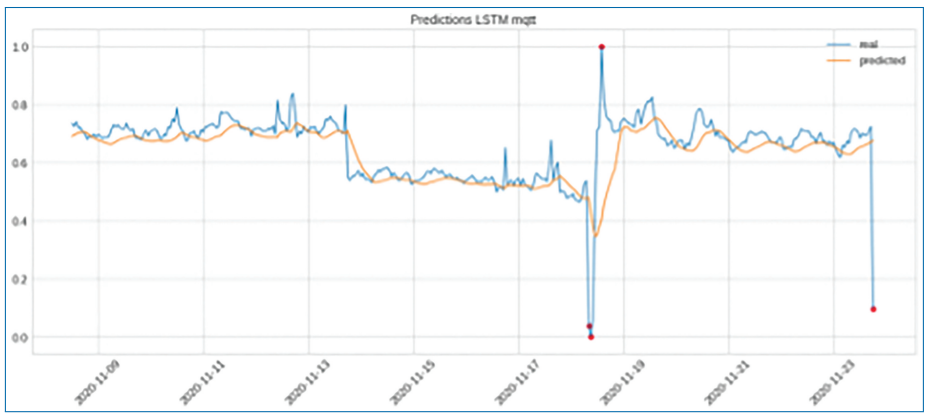


**Figure 7.9.** Vanilla LSTM predictions for MQTT protocol in the Smart Building.

Considering that each SIS can be completely different in terms of diversity of network protocols in the communications, type and number of devices, type and amount of operational data exchanged, etc. the anomaly detection capability of the enabler must be adjusted to each type of SIS, which means that the detection models need to be re-trained for each SIS.

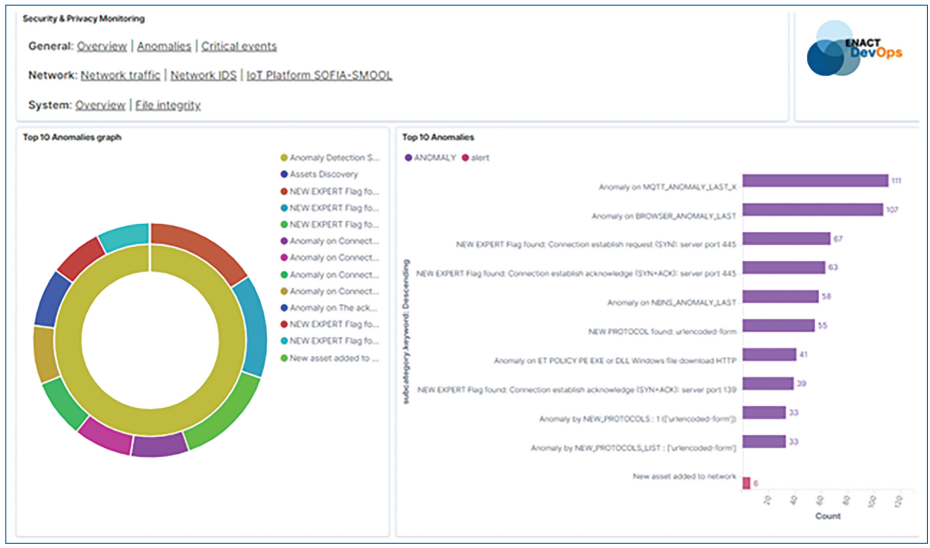


**Figure 7.10.** Extract of the overview dashboard view in the Smart Building.

The security administrator or SIS operator can continuously be aware of the cybersecurity status of the SIS using the front-end of the enabler, which displays all SIS status data, prediction data and detected security events in a user-friendly manner in form of alerts, statistics and graphs. Within ENACT, multiple viewpoints have been implemented in the front-end in order to have a comprehensive overview of the SIS security status; additionally, each of the viewpoints includes many dashboards. Nevertheless, the front-end can be adapted ad hoc in case the end user wants to have more details, or customize the graphs and the rest of the visualization objects.

Figure 7.10 shows an extract of the overview dashboard view of the General viewpoint in the enabler implemented for the Smart Building System use case (cf. Chapter 11). It offers the most important information related to the security events generated over the SIS to protect. The end user can navigate through the rest of the dashboards to learn more details about security events.

Figure 7.11 shows the network traffic dashboard of the Network viewpoint and Figure 7.12 shows an extract of the anomalies dashboard of the General viewpoint. Both dashboards have also been customized for the Smart Building System use case.

The Security and Privacy Monitoring and Control Enabler has been designed with the capability to integrate with an IoT platform, specifically with the SMOOL

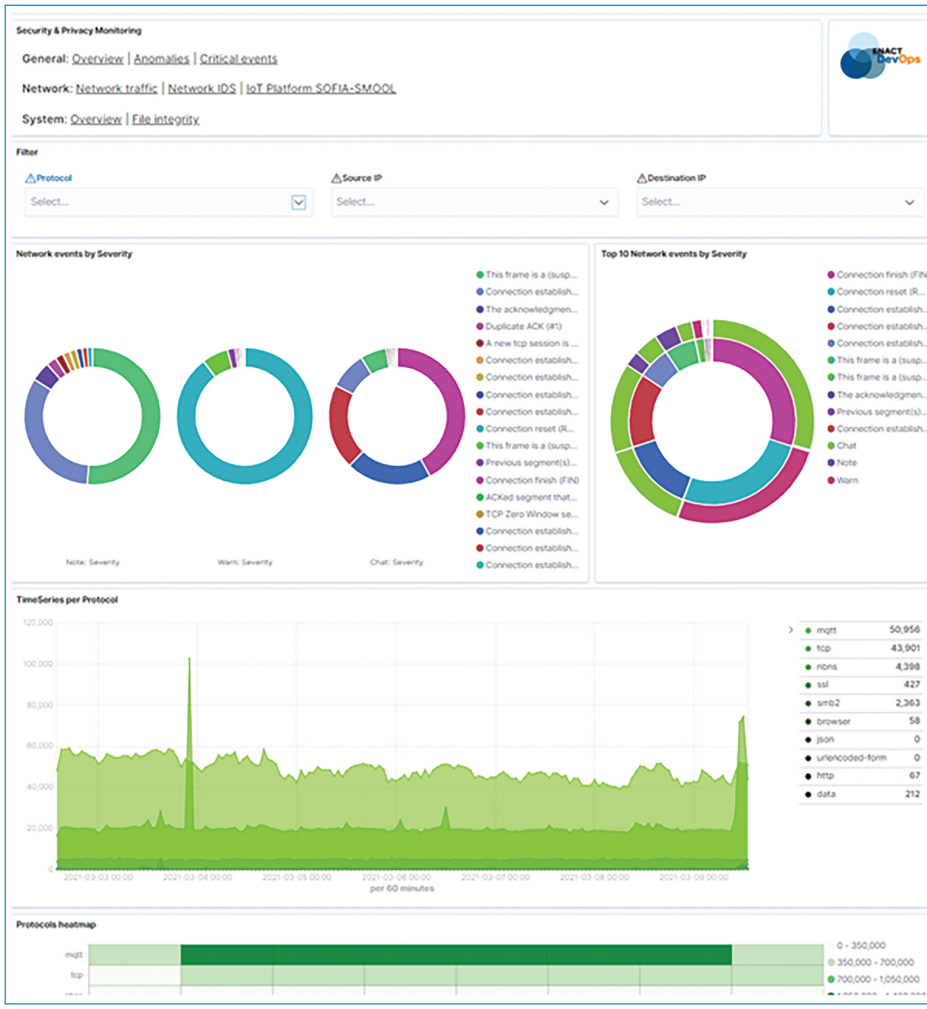


**Figure 7.11.** Extract of the network traffic dashboard of the Network viewpoint in the Smart Building.

IoT Platform (cf. Chapter 7.2). This particular implementation allows the security administrator to continuously monitor all communications and data exchanged through the IoT Platform. Figure 7.13 shows the architecture of the enabler integrated with SMOOL IoT Platform.

The distinctive feature in this scenario is that a client of the SMOOL IoT Platform, called Security KP (cf. Chapter 7.2.2), works as an app agent for the enabler by monitoring all data and communications through the IoT Platform and identifying

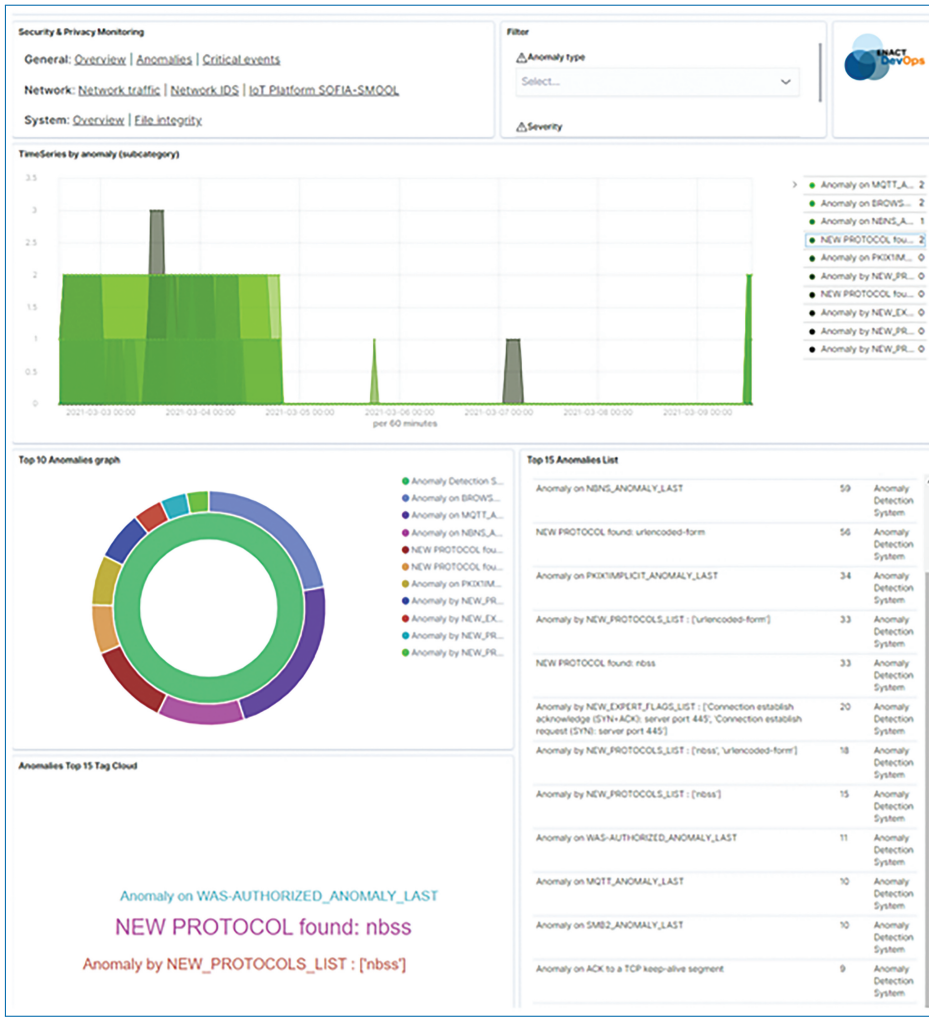


**Figure 7.12.** Extract of the anomalies dashboard of the General viewpoint in the Smart Building.

potential malicious intrusions. Furthermore, the integration with SMOOL IoT Platform provides the enabler security control capability to react to security incidents. For example, when detecting an unauthorized communication within SMOOL IoT Platform by the app agent (i.e. the Security KP), the enabler can respond by blocking all communications coming from the unauthorized client. All these security events registered by the SMOOL IoT Platform are shown in the front-end of the enabler.

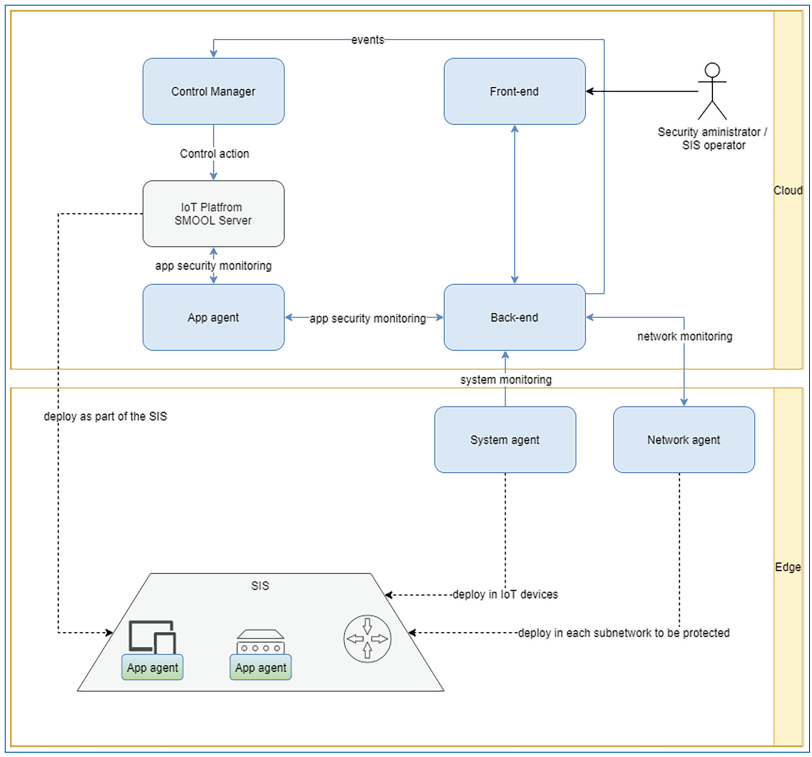


**Figure 7.13.** Architecture of the the Security and Privacy Monitoring and Control Enabler integrated with SMOOL IoT Platform.

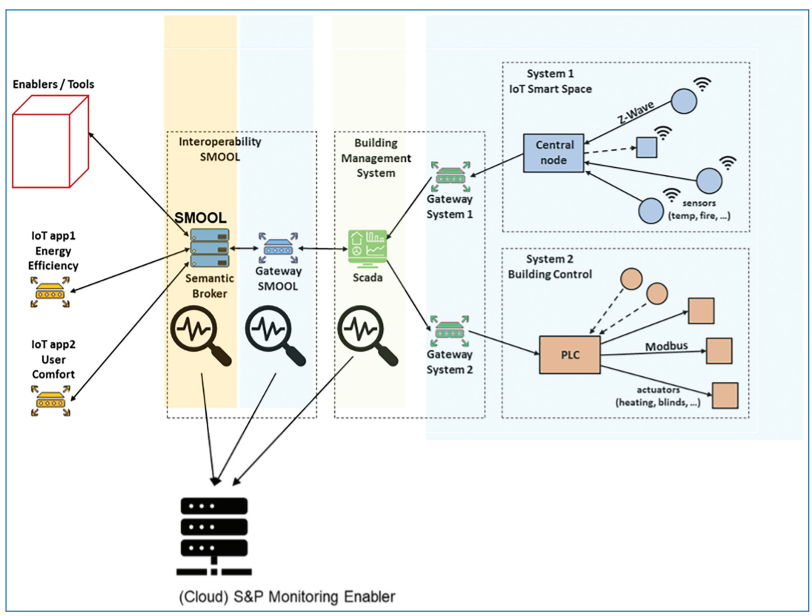


**Figure 7.14.** Smart Building SIS high-level architecture with the Security and Privacy Monitoring and Control Enabler integrated.

### 7.3.2 Validation

The Security and Privacy Monitoring and Control Enabler has been validated in two different use cases in the ENACT project.

#### 7.3.2.1 Smart Home System

The Smart Home System (cf. Chapter 11) has integrated the Security and Privacy Monitoring and Control Enabler together with the SMOOL IoT Platform in order to monitor the IoT applications of user comfort and energy efficiency of the building. In that way, the Smart Home System has been monitored at different layers: network layer covered by network monitoring agents, system layer covered by system monitoring agents in IoT devices such as Raspberry Pis, and app layer covered by SMOOL KP clients as app agents.

The SIS operator of the Smart Building System has been able to discover anomalies and security incidents by using the enabler. For example, Figure 7.15 shows an anomaly in the network protocols used by the Smart Building system related to HTTP protocol's content type formats (such as json, image-gif or png); this

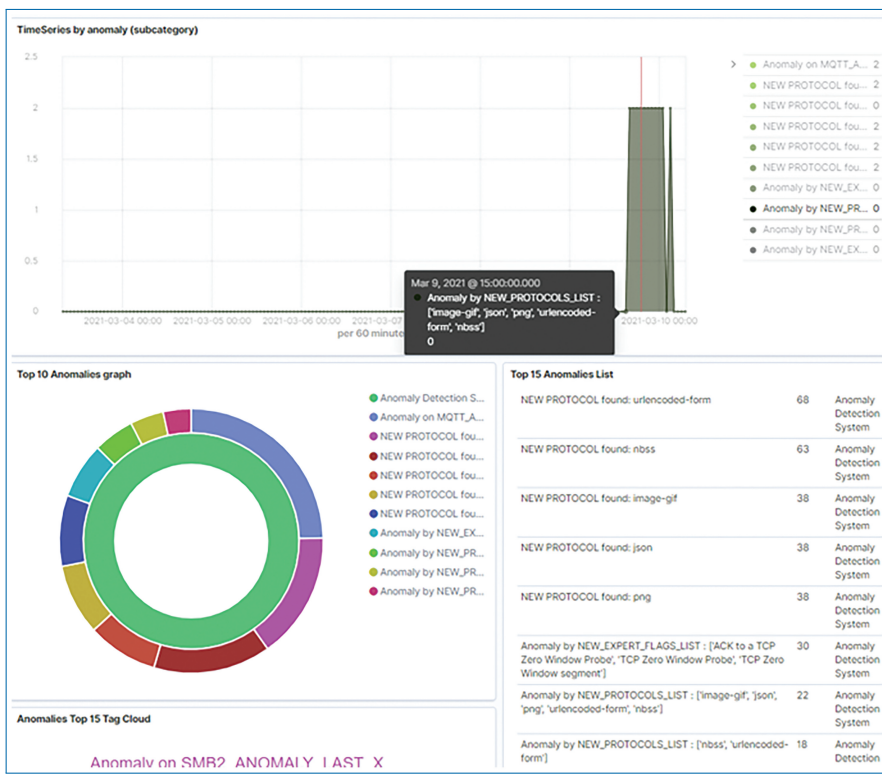


**Figure 7.15.** Example of network anomaly detection in the Smart Building.

anomaly can be seen in different graphics of the dashboard: (i) in a timeseries graphic or (ii) in a top 10 anomalies list.

#### 7.3.2.2 Intelligent Transport System

The Intelligent Transport System (cf. Chapter 10) has integrated the Security and Privacy Monitoring and Control Enabler in order to monitor the security status of the on board Edge part of the train system. The enabler has been used and validated for the following scenarios:

1. User and entity behaviour monitoring, which is mainly based on the anomaly detection capabilities offered by the enabler. Industrial protocol network traffic has been analysed in order to detect potential security incidents related to abnormal traffic behaviour. When an anomaly is detected, the enabler is able to react by enabling a specific security control for the SIS itself.
2. Intrusion detection, which uses rule-based detection capabilities implemented within network and app agents of the enabler to spot the unauthorized users and devices trying to communicate or get access to resources in the system.

## 7.4 Context-aware Access Control

### 7.4.1 Purpose

Access control and identity governance mechanisms are cornerstones of security and privacy, which is today focused on addressing people accessing IT applications. In the context of the Internet of Things, access control needs to be extended to address not only people accessing IoT, but also to manage the relationships between connected things. This requires designing and building new access control mechanisms for authorizing access to and from connected things, with ad hoc protocols while still being able to address traditional access to IT applications.

The key challenge for access control in IoT is dynamicity. IoT systems are changing continuously: Devices keep entering and exiting the system; The same devices may be used in different context; New connections emerge among the devices; etc. For such highly dynamic IoT systems, access rights from people to devices, and from devices to devices, are not immutable. The access rights may vary according to the context change. Take an eHealth scenario as an example, where senior adults use IoT devices to monitor their physiological data such as blood pressure. In the normal, day-to-day context, only the user himself should have the access right to the data, due to the privacy concern. However, in a special context, such as under

emergency rescue, medical staff should be granted with the access right to the journal with historical physiological data. Therefore, the decision of access right in IoT systems must be made with awareness of the context.

The objective of the Context-aware Access Control (CAAC) is to deal with these considerations, by providing dynamic access control mechanisms for IoT systems based on context awareness and risk identification, applicable to both IT (Information Technology) and OT (Operational Technology) domains, through an IAM (Identity and Access Management) gateway for IoT that includes next-generation authorization mechanisms.

Evidian Web Access Manager (WAM) provides security features for identity management and access control. The Context-Aware Access Control tool is an evolution of the authentication and authorization mechanisms provided by WAM intended for the Internet of Things.

### 7.4.2 Background: Industry Standards of Access Control Protocols

- **The traditional dynamic access control chain based on the XACML model**

  A first approach is to study how the traditional dynamic access control chain based on the XACML model [10] could help to answer the challenge of securing the Internet of Things.

  XACML is a policy-based management system that defines a declarative access control policy language implemented in XML and a processing model describing how to evaluate authorization requests according to the rules defined in policies. As a published standard specification, one of the goals of XACML is to promote common terminology and interoperability between authorization implementations by multiple vendors.

  XACML is primarily an Attribute-Based Access Control (ABAC) system, where attributes associated with an entity are inputs into the decision of whether a given entity may access a given resource and perform a specific action.

  The XACML model supports and encourages the separation of the authorization decision from the point of use. When authorization decisions are baked into client applications, it is very difficult to update the decision criteria when the governing policy changes. When the client is decoupled from the authorization decision, authorization policies can be updated on the fly and affect all clients immediately.

  The access control chain based on the XACML model is depicted in Figure 7.16.

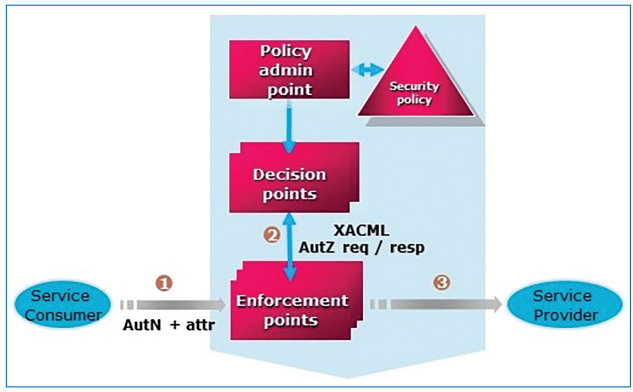


**Figure 7.16.** The dynamic access control chain based on the XACML model.

In this chain:

- The Policy Decision Point (PDP) evaluates access requests against authorization policies before issuing access decisions.
- The Policy Enforcement Point (PEP) intercepts the user's access request to a resource, makes a decision request to the PDP to obtain the access decision (i.e. access to the resource is approved or rejected), and acts on the received decision.

In fact, this approach is dynamic by essence, since the access control decisions are made based on attributes associated with relevant entities. In addition, it offers a powerful access control language with which to express a wide range of access control policies.

But the following points make this approach prohibitive:

- An approach based on rules is difficult to administer. Defining policies is effort consuming. You need to invest in the identification of the attributes that are relevant to make authorization decisions and mint policies from them. In addition, the ABAC system introduces issues, most notably the 'attribute explosion' [3] issue and, maybe more importantly, the lack of audibility.
- Although Service-Oriented Architecture and Web Services offer advanced flexibility and operability capabilities, they are quite heavy infrastructures that imply significant performance overheads.
- Since XACML has been designed to meet the authorization needs of the monolithic enterprise where all users are managed centrally, this central access control chain is not suitable for cloud computing and distributed system deployment, and it does not scale to the Internet.

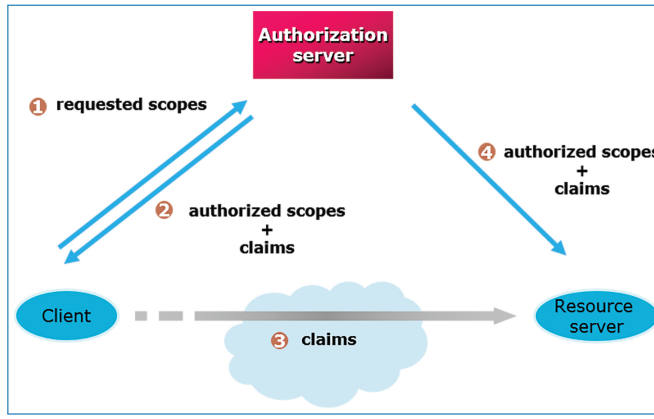


**Figure 7.17.** Another approach based on OAuth 2.0.

- **Another approach based on OAuth 2.0**

Another approach has been studied, based on the OAuth 2.0 industry-standard protocol for authorization [5].

This approach is depicted in Figure 7.17.

In this approach, a client can access a resource on behalf of a user through an authorization delegation mechanism. This assumes that the user has given his consent for the requested scopes.

As a major advantage, this protocol can be implemented in a light way, by leveraging HTTP and REST-based APIs. In fact, OAuth 2.0 supports the mobile device application endpoint in a lightweight manner. Its simplicity makes it the de-facto choice for mobile and also non-mobile applications. Due to the growing importance of Cloud technologies and APIs, the REST architecture is now heavily favoured.

In addition, OAuth 2.0 allows a fluid integration with role management: OAuth 2.0 scopes can be used to provide role-based authorization.

But this protocol does not have the granularity of XACML in terms of rules. And another point is still an obstacle to meet the need of an IoT context-aware access control: the dynamicity, allowing to take into account the context, is not provided by design.

### 7.4.3 A Solution for a Context-aware Access Control Approach for IoT

Due to the disadvantages observed on the traditional dynamic access control chain based on the XACML model, it appears that a solution based on OAuth 2.0 is more appropriate.

But to provide an IoT context-aware access control mechanism, the gap must be filled to deliver dynamic authorizations based on context by using the OAuth 2.0 protocol.

Starting from security features for identity management and access control based on the protocols OAuth 2.0 and OpenID Connect (OIDC) [11], the approach is to develop an evolution of these authentication and authorization mechanisms intended for the Internet of Things. Due to the dynamic nature of the data regarding the environment of the connected devices and the persons they belong to, this contextual information must be used to manage and adjust the security mechanisms, i.e. consider contextual information in the identification of the entity requesting access and in the evaluation of the conditions to grant access.

By assessing the applicability of OAuth 2.0, the ENACT IoT context-aware access control leverages it as a key protocol for interoperability, by adding dynamicity to the authorization decisions produced by OAuth 2.0, although this was not originally intended in that protocol. This dynamic capability is in charge of taking contextual information into account and inserting it into authorization decisions.

### 7.4.4 Architecture

The Context-aware Access Control tool provides an authorization mechanism that issues access tokens to the connected objects after successfully authenticating their owner and obtaining authorization. An access token contains the list of claims and scopes that an authenticated user has consented for this object to access. These scopes and claims are used to restrict accesses to the back-end server APIs to a consented set of resources.

This authorization mechanism may be coupled with contextual information to adapt the access authorizations according to them (for example to make certain information more widely available in some urgent case).

To this objective, the Access Control Tool directly communicates with a Risk Server to make dynamic access controls based on the context information during the authorization phase. For example, it can reject the authorization if the access token is valid while other context information does not respect the authorization policy.

The authorization policy is a set of rules that define whether a user or device must be permitted or denied access to a back-end server. An administrator can control this adjustment and create special authorization rules based on the context data provided.

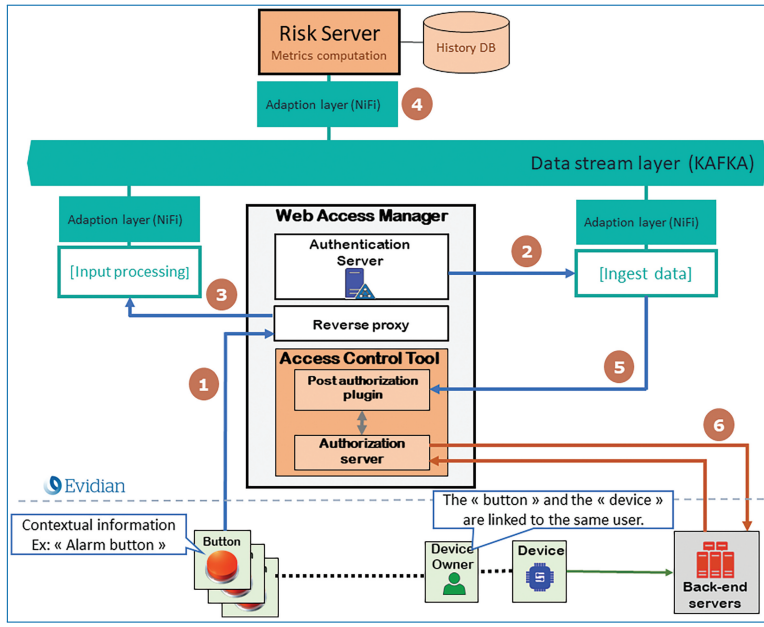


**Figure 7.18.** Context-aware Access Control global infrastructure.

The Context-aware Access Control tool is provided inside an infrastructure aimed to gather contextual information to deduce a risk level associated with a user. Figure 7.18 gives a global view of this infrastructure.

The infrastructure is based on the Apache Kafka event streaming platform, which allows to publish and subscribe to streams of events. The principle is to publish in this platform contextual information which may come (1) from connected devices (sensors, alarms, etc.) or (2) from audit events produced by the Evidian Web Access Manager. The contextual information is sent to an input processing interface (3) which then publishes it to a Kafka topic. An event is then received by the risk server from an Apache NiFi interface (4) which will take into account the contextual information in a dynamic risk level computation. Then, when a device tries to access a resource, (5) the CAAC retrieves the dynamic risk value associated to the device owner, and (6) this is transmitted to the back-end server to modulate the access accordingly.

In this architecture, two components are providing the Context-aware Access Control mechanisms:

- The **Access Control Tool**, composed of an Authorization server associated to a Post authorization plugin, to add more controls during the authorization phase. Its purpose is to check if the request is authenticated and is authorized to access the back-end server.

Indeed, each time a device sends a request to a back-end server (7), WAM can check the dynamic claims and scopes consented by the user associated to the device that performs the request and, in turn, realize special actions according to this information such as blocking the request or limiting the accessible resources.

The Post authorization plugin extends the basic authorization phase and is entirely customizable. Any operation can be executed during the authorization phase, including calling external programs, and in particular the Risk Server. The Post authorization plugin can create injection variables that can be reused and injected in the initial request sent to the back-end server.

- A **Risk Server** which essentially relies on WAM audit events to calculate a risk level for each user. This allows detection of abnormal behaviour such as connections from unknown IP addresses, or multiple failed connections.

  A user's risk level is a function of the level of trust given to that user. This level of risk determines the level of trust that can be placed in the devices owned by that user. The user's risk level is based on a system of sanctions/rewards depending on the user's behaviour. Its computation uses a ranking system based on a user-specific score: the Risk Score.

  Contextual information coming from external sources (sensors, other applications, etc.) makes it possible to modulate this risk, i.e., to increase or decrease it depending on the situation. For example, in the case of a fire, a smoke detector immediately sends information to the Risk Server, which will considerably reduce the user's risk and allow easier access to resources.

  The contextual information is sent to an input processing interface which then publishes it to a KAFKA topic. For this contextual information transmission to be controlled and secure, the transmitting device must be enrolled in WAM and associated with a user, and it must have received a valid access token which allows it getting its owner's userid to be associated with the contextual information. Each device is associated with a risk factor which can be used to modulate the user's risk score.

### 7.4.5 Integrating the Context-aware Access Control Tool

- **Device enrolment**

  The device enrolment procedure allows a device to be associated with the identity of its owner. The Access Control tool leverages on the OAuth 2.0 Device Flow protocol to achieve this.

  The only requirements to use this flow are that the device is connected to the Internet and able to make outbound HTTPS requests, and that it is

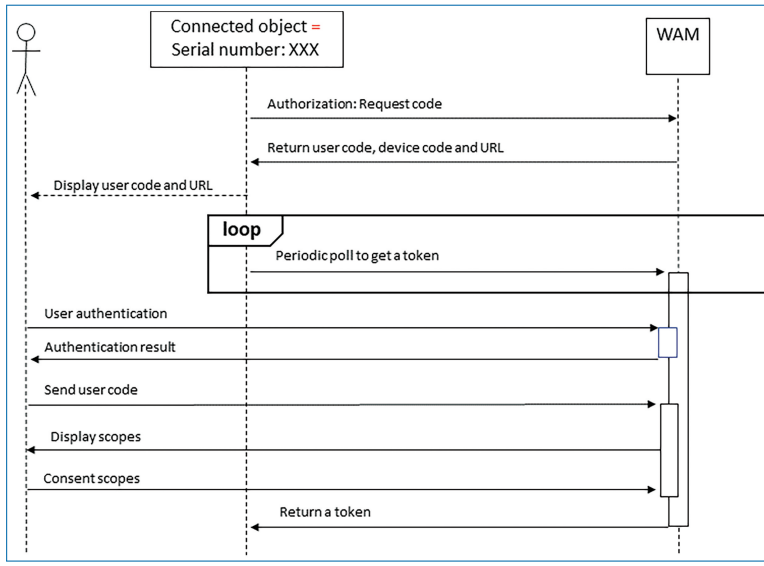


**Figure 7.19.** Device enrolment sequence diagram.

able to display or otherwise communicate a URI and code sequence to the user, and that the user (device owner) has a secondary device (e.g., personal computer or smartphone) from which to process the request. There is no requirement for two-way communication between the OAuth client (i.e., the connected device) and the end user's user-agent, enabling a broad range of use-cases.

During this procedure, the user gives his consent to the device to access data scopes on static attributes (username, email, etc.) and also a dynamic attribute (a risk level computed from contextual information on the user). At the end of the enrolment phase, the device receives an access token. The device has now access to the device owner profile that includes static attributes (username, email, etc.) but also the dynamic risk level.

The sequence diagram for this device enrolment procedure is described in Figure 7.19.

- **Context-aware Access Control with WAM used as reverse-proxy**

In this case, WAM is used as a Reverse proxy to protect the back-end servers. Additionally, WAM checks the token of the incoming request to verify if the device is authorized to access the back-end server. If this is the case, WAM injects in the header of the initial request the consent scopes of the device owner. This injection does not modify the request and the scopes injected contain some information about the device owner (username, email, etc..) and a risk level computed from contextual information. This allows the

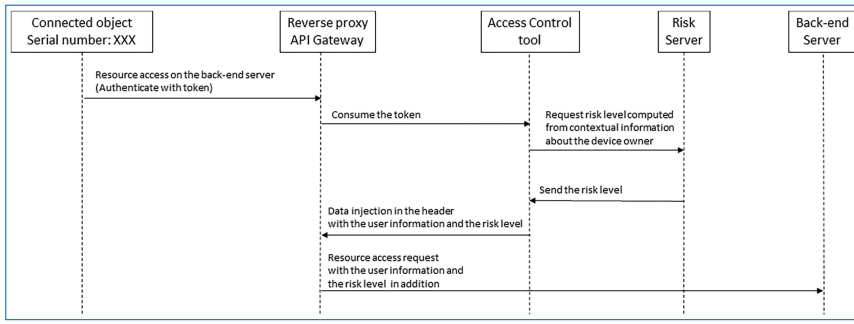


**Figure 7.20.** Context-aware Access Control with WAM as reverse-proxy — Sequence diagram.

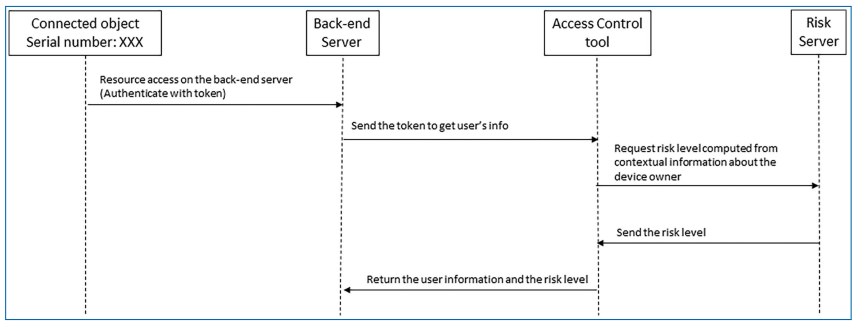


**Figure 7.21.** Context-aware Access Control with WAM as OpenID Connect IDP — Sequence diagram.

back-end server to make the link between the requesting device and the user associated with it, and to know the risk level depending on the context.

The sequence diagram for this working mode is described in Figure 7.20.

- **Context-aware Access Control with WAM used as OpenID Connect IDP**

In this case, WAM is used as an OpenID Connect IDP. WAM handles the access control part by checking if the token sent to a back-end server is valid. If this is the case, WAM responds with the consent scopes of the device owner. These scopes contain some information about the user (username, email, etc.) and the contextual risk level.

In the case where the authorization policy is not respected, the Access Control tool will inform the back-end server that it has to reject the request made by the device.

The sequence diagram for this working mode is described in Figure 7.21.

The device uses its token to access the back-end server (for example to push some data). The back-end server checks the validity of the token and retrieves the device owner's consent scopes for this token by calling the user-info endpoint of WAM (the userinfo endpoint from the Access Control tool

API consumes a token to retrieve information on the user). WAM returns the user information (for instance: username, email, address) and the dynamic contextual risk level associated to the user. The back-end applications can use this additional information to perform special actions.

### 7.4.6 Main Innovation

The main innovation brought by the features offered by the Context-Aware Access Control Enabler can be summarised as follows:

- The solution provides one unique tool to control in the same way the access of all the IoT actors (end-users, services, devices, administrators) to the operated data and resources, for both IT and OT (operational technologies) domains.
- The solution adds dynamicity to the authorization decisions OAuth 2.0 produces, by injecting dynamic scopes in the standard device flow.
- This allows to exploit contextual risk levels as dynamic attributes in the authorization mechanisms.
- Accordingly, the provided authorizations can be adapted based on a risk level computed from contextual data on the user and his devices, which allows context-aware dynamic access control behaviors.

## 7.5 Conclusion

This chapter was dedicated to the Security and Privacy Monitoring and Control Enabler designed to be used at the Ops phase of the DevOps life-cycle of SIS to address the security aspects of trustworthy SIS operation. The enabler is an innovative solution that supports SIS operation with multi-source data capturing, advanced detection combining signature-based IDS and AI techniques, and comprehensive situational awareness through a rich multi-viewpoint dashboard. By using this enabler, it is possible to assemble all or only some of the components in the enabler architecture. This brings flexibility to the continuous monitoring since it is possible to tailor the enabler design to the particular needs of the SIS under study in terms of e.g., how many security agents are deployed and where, which security policies are used by the clients, which metrics are monitored, and the needed tailored alarms and data visualisations can be created ad hoc.

The continuous monitoring offered is holistic in the sense that it correlates data captured in the three main layers of the SIS: network, system and application layers. And this makes possible a high richness and accuracy of security incidents

and anomalies detection which leverages signature-based detection together with machine learning and deep learning-based detection.

The enabler also answers to the needs of rapid elasticity and full scalability required by SIS that involve large amounts of sensors and actuators, while it still is able to offer the required visualisations and notifications that constitute the basis for the informed situational awareness of the overall system.

In order to be able to take advantage of the insights gained by the tool over the SIS, the enabler was designed with a security event bus for integration with other cybersecurity threat intelligence platforms and services, such as those of forensics analysis and cybersecurity information sharing with third parties.

Last but not least, the enabler design permits a seamless integration with controls at application layer, for example those developed on top of the SOFIA SMOOL IoT platform monitoring and control agents' management, which are able to monitor and control secure communications among the smart things of the SIS.

As part of the future lines of work in the enabler, the automatic reaction capabilities will be researched and enriched by extending the controls with intelligent security orchestrators and decision making support that facilitate the combination of multiple reaction measures when needed in the different layers of the SIS, so as the security level of the system is increased in the face of attack or incident symptoms.

## References

[1] Rego *et al. SMOOL source code*. https://bitbucket.org/jasonjxm/smool, 2011–2020.

[2] Elasticsearch B.V. *The Elastic Stack. Retrieved March 11, 2021*, 2021. URL: https://www.elastic.co/elastic-stack.

[3] Prosunjit Biswas, Ravi Sandhu, and Ram Krishnan. *Attribute Transformation for Attribute-Based Access Control*. https://profsandhu.com/confrnc/misconf/abac17-prosun.pdf, 2017.

[4] Apache Software Foundation. *Apache Kafka. Retrieved March 11, 2021*, 2017. URL: https://kafka.apache.org.

[5] Dick Hardt. *The OAuth 2.0 Authorization Framework*. 2012. URL: https://tools.ietf.org/html/rfc6749.

[6] Wazir Zada Khan *et al.* "Industrial internet of things: Recent advances, enabling technologies and open challenges". In: *Computers & Electrical Engineering* 81 (2020), p. 106522.

[7] NIST. *NIST Computer Security Resource Center Glossary. National Institute of Standards and Technology. Retrieved March 11, 2021*. 2020. URL: URL=%20 https://csrc.nist.gov/glossary/term/information_security_continuous_monitoring.

[8] NIST. *NIST SP 800-53 Rev. 5. (December 2020). Security and Privacy Controls for Information Systems and Organizations. National Institute of Standards and Technology. Retrieved March 11, 2021*. 2020. URL: https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final.

[9] Adrian Noguero, Angel Rego, and Stefan Schuster. "Towards a Smart Applications Development Framework". *Social Media and Publicity* 27 (2014). URL: https://bitbucket.org/jasonjxm/smool,%202011-2020.

[10] OASIS. *eXtensible Access Control Markup Language (XACML) Version 3.0. Retrieved July, 2019*. 2019. URL: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html.

[11] OpenID. *OpenID Connect*. 2014. URL: https://openid.net/connect/.

[12] Inger Anne Tøndel, Maria B. Line, and Martin Gilje Jaatun. "Information security incident management: Current practice as reported in the literature". In: *Computers & Security* 45 (2014), pp. 42–57.

[13] The Board of Trustees of the Leland Stanford Junior University. *Protégé, open-source ontology editor and framework.* 2020. URL: https://protege.stanford.edu/.