Providing Security to a Smart Grid Prosumer System Based on a Service Oriented Architecture in an Office Environment

Alexander Camek, Florian Hölzl, and Denis Bytschkow
fortiss GmbH,
Guerickestrasse 25, 80687 Munich, Germany
Email: {camek,hoelzl,bytschkow}@fortiss.org
http://www.fortiss.org/

***Abstract*— The transformation of energy systems into smart grids will provide a lot of new possibilities to all participants. By upgrading energy distribution nets with information and communication technologies (ICT) customers and suppliers will have additional information. It integrates a bi-directional information transport between both, power supply company and customer. This real-time data is used to save costs by reducing energy consumption, to provide a good alignment with fluctuating renewable energy production, or to actively participate in the market by offering surplus production. However, ICT systems are always a subject to potential cyber-attacks and manipulations. Therefore, securing such systems is one of the most important objectives. In this paper we conduct the security analysis of our smart grid prosumer demonstrator, which is implemented using on a service oriented architecture. We analyze the attackers' motivations and derive an attacker model. We identify weak points of our system with respect to the attacker model and the system architecture. Finally, we propose counter measurements to improve the case study smart grid system.**

***Index Terms*—Security Analysis, Attacker Model, Smart Grid, Prosumer, Office Environment, Service Oriented Architecture**

I. Introduction

The increasing integration of renewable energy sources creates new challenges for the existing electricity infrastructure. For example, it requires an increased attention and handling of volatile energy fluctuations in the power distribution nets. Upgrading the power system with information and communication technologies (ICT) is considered the right way to cope with these challenges. A first step is to integrate a bi-directional information transport between the power supply company and the customer. This data exchange provides the opportunity for load balancing, more efficient generation, and better computer assisted planning. Mechanisms, like demand side management and demand side response, tackles the road to an improved energy utilization.

Combining ICT and energy distribution networks led to the idea of smart grid systems. Classic power producers and consumers are supplemented by so-called *prosumers*, which are able to produce and consume electricity. Prosumer energy management solutions are complex systems with functionalities from building automation, agent-based market participation, control systems with real-time requirements, as well as, planning and optimization software. Therefore, modular and flexible approaches like service oriented architectures (SOA) in combination with design principles regarding real-time requirements are well suited to build such systems.

However, new energy management systems will only be accepted by the customer if the system functionality is reliable and secure. In this paper, we analyze a SOA-based prosumer system in the context of the smart grid technology with respect to its security aspects. We use the *Living Lab* demonstrator [1] implemented at the office building of fortiss as a case study. The demonstrator architecture is based on the SOA paradigm. This approach was chosen to allow adding new devices flexibly and improving the system with new services in a plug and play manner. However, such an open architecture is prone to attacks like data manipulation, data misuse, and privacy violations. Therefore, a thorough security analysis must be part of the development process of such systems. During the audit every component and their gaps were recorded for deeper analysis. Then, we deduced matching counter measurements to close these gaps. Most of the problematic gaps and our counter measurements are closely related to our demonstrator. Our experience can be used as a blueprint to provide security to other SOA-based prosumer systems.

The paper is organized as follows. In Section II we start our security analysis of the prosumer demonstrator by examining the motivations of attacks against a smart grid system in general. In Section III we derive an attacker model from these motivations. Section IV presents the Living Lab demonstrator and its architecture. Next, we will analyze the system in Section V and propose countermeasures in Section VI. Section VII concludes the paper.

II. Motivations to attack the Smart Grid

Attackers can have different motivations against a common smart grid. An example of a simple attack is the theft of equipment or devices.

Consumption of electricity without payment or minimizing the bill is a possible motivation. Every smart grid component, especially a node like an office building, is connected to the

TABLE I
ATTACKER CLASSIFICATION

Skill Class	Class A	Class B	Class C	Class D
Smart grid knowledge	medium	high	high	medium
IT knowledge	low	medium	high	high
Technical equipment	low	medium	medium	high
Financial resources	low	low	medium	high
Possible roles	house owner, petty criminal, or script kiddie	manufacturer, service people, or tweaker	device developer, service people, or hacker	terrorist or organized crime

smart grid, which is controled by power system operators providing electricity when needed. An attack can be done by manipulating the internally connected smart meter to reduce the payment or by interfering with the communication between prosumer and Power Company. An advanced communication attack is a manipulation of the power supplier's smart grid by manipulating the prosumer.

Another motivation is based on the business model where a supplier or equipment manufacturer has equipped devices with additional functionality only usable by extra payment. An attacker tries to use this additional functionality by unlocking the specific or dutiable functionality without this payment. Even using additional hardware functionality of devices, which are not supported by the runtime software, can be a target.

Reuse of stolen devices or adding self-crafted components is also a motivation for an attack. Stolen devices may have to be reprogrammed to fit into the new environment, but the effort to do this must be less than the cost to buy the original part. By adding self-crafted components the user wants to solve a specific problem, for which there is no solution at the free market. These unauthorized components can harm the smart grid by sending wrong data, influencing other system components, or hindering other components' operation. Thus, this kind of self-crafted components can be used to attack key material, especially to read or to change it, or can be part of a masquerade attack where these component claim to be a specific instance with a unique identifier. This is a kind of authorization violation, because every component has to be verified to be compatible with other components. Besides using self-crafted components, a user could manipulate a given hardware, manipulate the component connections by disconnecting it, reverse engineering a given hardware, read out memory, or reprogram the hardware.Simply put, a malicious user could exchange a device with a manipulated one.

Further attacks are not directly targeted against the usage of electricity, but against the prosumers's sensitive information. The exchanged data between a customer and the power company is interesting for attackers. An attack is most often attempted by a man-in-the-middle attack. Here, an attacker will wiretap the communication between both parties. He tries to access sensitive information, such as credit card data, personal data of a customer, data of a power company, and data of subsystems. When somebody wiretaps a communication he could also change the communicated data. This enables the attacker to manipulate the purchased amount of electricity, change account data, or even harm the entire system. Thus, the manipulation is not only done by changing the data, but also by deleting some specific information or to replay recorded data in future.

Some attackers, in particular those without malicious intents, want to manipulate the system in order to find security holes and to help closing them. By exposing security holes these attackers want to increase their reputation, e.g., in the scientific community.

Harmful attackers will manipulate the internal system not only to destroy equipment, but also to injure users. Another severe attack in this class of motivations is extortion. The attacker tries to extort a smart grid operator, a supplier, or a device manufacturer by threatening to uncover some detail of the system, which is valuable to them.

Finally, intellectual property is a very well paid good for competitors or new suppliers and, therefore, its receipt is a motivation. Device manipulation with the goal of analyzing internals of the system is a way of attack in this respect.

Of course, the motivations given here are not a complete list of possible attacks against a smart grid system, but were selected from the perspective of our use case.

III. ATTACKER MODEL

Based on the motivation given in the previous section we now classify possible attackers into different categories. Such categorizations have been studied in related work.

Howard and Longstaff [2] provide a common model to describe computer security incidents. Here, an incident is a relation of an attacker to an objective; the latter is comparable to our motivations. Their seven attacker categories are primarily defined from the perspective of computer and network security. In contrast to our categorization attackers are not divided into internal and external physical access to the system.

In [3] attackers are categorized according to their physical access to the target and their skills. Chinnow et al. [4] classified their attackers based on the physical access to the smart meter, e.g., the billing component between the Living Lab and the power supplier. They distinguish between outsiders (no physical access) and insiders, which are classified into homeowners and energy provider employees.

Based on the given motivations we classify our attackers into four skill classes, as given in Table I:

1) Class A: includes technically unskilled persons searching for a low-risk opportunity to steal and sell some devices, or to attack and change systems. This attacker possesses low technical equipment and uses by experts pre-crafted tools. The members of this category have only minor financial resources. The house owner is part of that category. House owners normally know what kind of equipment is installed and how it is organized. They can use their knowledge to improve the usage of the system. We also add the petty criminal to this category. He normally steals devices and wants to sell these. Script kiddies [5] who attack systems only for fun without special knowledge fit in this category. They want only to improve their reputation, but their attacks will harm the system and after those attacks the system is left in an unusable mess.

2) Class B: contains technically low skilled persons who have already attacked systems with weak security, and may have been caught for one of their attacks. This kind of attacker is cunning and experienced. The person only attacks smaller systems or systems with weak security. They possess some technical equipment, where special tools are part of. The members of this category have only minor financial resources. Device or system manufacturers are part of that category. We also add service people to this category, because they know about the system and possess the needed special tools. In this category we also put tweakers who try to improve devices by changing the software or added hardware.

3) Class C: covers technically high skilled persons who are specialized in attacking systems. Sometimes they use members of our other classes as a help. We put normal developers of devices or infrastructure in this category. Sometimes these people want to increase their reputation or learn something new, known as hackers. But mostly these people attack for money or harming the company. Additionally, service people who are paid by house owners to tune or tweak the system are also part of this category. They know the system and can use insider knowledge. They have the tools and special equipment to perform the attack. The goal of this attacker class is to use their knowledge to improve their personal benefit.

4) Class D: includes technically high skilled or trained persons who have substantial resources. Additionally, the attacker uses members of our other classes as help. One can interpret this category as terrorists or mafia members. We define this category for criminal organizations. The goal of this attacker class is to do a maximum of impact by harming the prosumer, and getting the highest possible outcome, such as money or terror.

Furthermore, we distinguish between *restricted* and *unrestricted* physical access of the attacker (Table II). An attacker with unrestricted access to smart grid devices can manipulate, replace, or remove them directly. Attackers with restricted access to those devices can manipulate the system only through the ICT infrastructure, i.e., they can interfere with the normal operation or even shut it down.

TABLE II
PHYSICAL ACCESS

Physical Access	unrestricted	restricted
Possible roles	house owner, tweaker, petty criminal, manufacturer, device developer, or service people	script kiddie, hacker, organized crime, or terrorist

IV. CASE STUDY

In order to evaluate the threats to all smart grid concerning parties and identify counteractive measures we will focus on a prosumer system based on SOA principles. SOA systems are currently used in several research demonstrators. An overview of different smart energy management systems and details of the implementation of our *Living Lab* demonstrator is provided in Koss *et al.* [1]. This system is representative for a smart grid system in an office building with its facility owners, its employees, and its service assistants. The demonstrator is an ideal case study, since it is prone to a large variety of attack scenarios. In order to provide a consistent security analysis in this paper the hardware setup and system architecture is summarized next.

A. Hardware Platform

The *Living Lab* prosumer system is integrated into an office building.It is equipped with four different types of devices as shown in Figure 1.

Energy production and storage components manage the self-produced electricity. A photovoltaic system on the roof produces up to 5 kW power. Batteries store that energy and provide the possibility for a set of offices and a meeting room to run in independent mode for several hours.

Sensors and actuators, typically found in the domain of building automation, provide data about the environment and enable user interactions. They use wireless communication based on the EnOcean technology.

Smart meters are connected via IP-based communication and provide a stream of data about the power consumption and grid properties, such as the power flow, the power factor, and the current frequency.

The last group of devices is used for thermal management. They control the heating system and, for example, the air-conditioning system in the server room.

Sensors and actuators send information about their status as events to the backbone software system. The events are stored in a data base and analyzed by server applications to carry out certain actions. User interfaces visualize the system status to users and can be used to trigger actions, such as turning lights on and off. Depending on the user and his role (e.g., employee or facility manager), the user interface provides only restricted options. For example, an employee can only control the devices in the room, which is assigned to him.

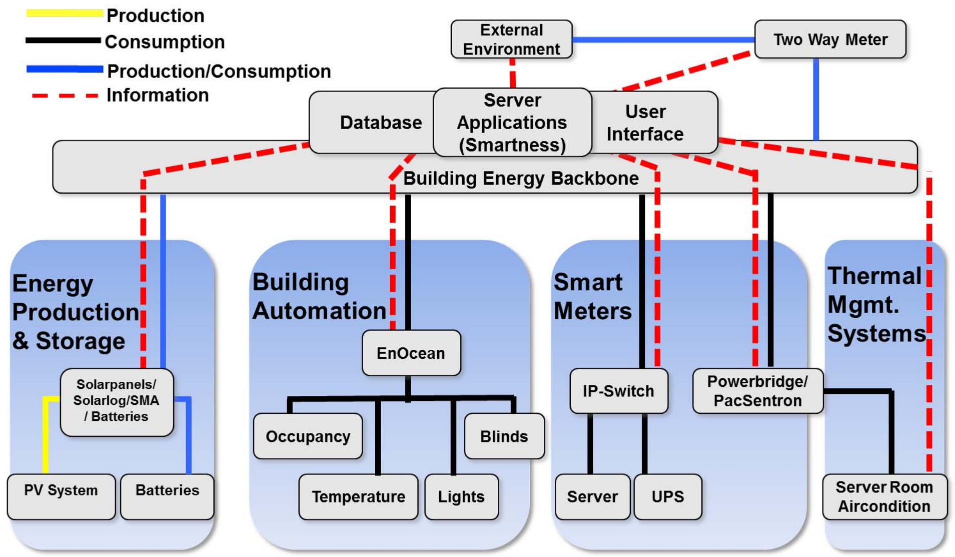

Fig. 1. The *Living Lab* components with their information and energy flow

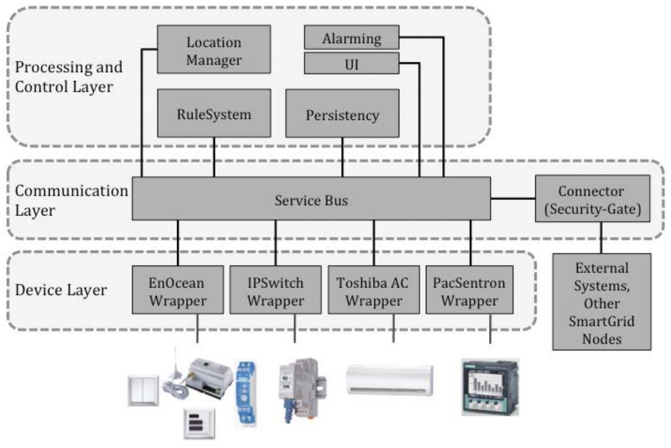

Fig. 2. The service-oriented architecture of the *Living Lab*.

B. System Architecture

As depicted in Figure 2 the system architecture has three distinct layers encapsulating different functionalities.

The *Device Layer* serves as the connection between hardware devices and the software layers. Wrappers are used to communicate with devices and translate device-specific and proprietary protocols to a common internal system protocol. New devices can be attached in a plug-and-play manner to the system by providing suitable wrappers.

The *Communication Layer* provides the messaging mechanism within the system. The demonstrator uses Apache ActiveMQ to implement this layer, which again uses the Java Messaging Service and the SOA protocol for message encoding. To increase the performance of the system the communication is currently based on events.

The *Processing and Control layer* provides the advanced functionalities of the system. The persistency services stores the system status data, which is used by the rule system to decide appropriate actions. The rule system implements a large part of the smartness of the system. It allows the configuration of automation rules, such as automatically turning off the lights when no one is detected in the room. More advanced rules can be created by the users with their respective permissions. The location manager serves to represent relationships between components of the system, such as sensors and actuators.

V. Security analysis of the Living Lab

In the previous section we introduced our case study system and its architecture. We explained the used hardware and their specific roles in the system. The architecture defines each software element and their interconnections. As the next step we analyze our use case with respect to possible security breaches based on our attacker model. Therefore, we described all possible attacks with attack trees [6], [7]. For simplicity we omit our attack trees and describe only the found attacks in close relation to our system architecture.

A. External weak points

In our office environment there exists only two connections from the Living Lab system to the outer world. The costs of defending the system against Class D attackers outweigh the benefits. Therefore, we take into account only Class C attackers with restricted physical access (Table II).

The two way meter in Figure 1 connects to the outer grid and deals with the payment. An attack against this system will harm the connection to the power supplier. When the two way meter is under the control of an attacker it is possible to provide wrong data to the system. This will lead to wrong measurements, and then cause wrong control commands at dependent subsystems.

A second external security problem is shown in Figure 2, where a *Connector* coordinates the communication to external systems. By this, we allow other external systems to query and read measurements given by our office environment. This connection will allow an attacker to enter and attack the system. Eventually an attacker could also hinder others to read measurements by an denial of service attack (DoS).

B. Internal weak points

In the previous section we used only the Class C attacker from our model. Here we apply Class A, B, and C attackers with unrestricted physical access. As internal attacks we exclude social engineering and physical access, such as cutting wires, rearranging connections, removing hardware, destroying hardware, and vandalism. We assume our system as operationally unsafe when an attacker accesses our system. With this condition we start our analysis at the lowest level in the system architecture (Figure 1).

At the Device Layer we use sensors to collect data and actors to complete commands. Sensors and actuators are mostly small devices with low CPU power and memory amount. Additionally, most of these devices are powered by low energy supplies, e.g., batteries or energy harvesting. These devices only support weak authentication schemes and simple cryptographic mechanism, due to low computation and power support. This allows an attacker to integrate a self-crafted device and, if necessary, masquerade it as an original one. To avoid that, the security of these devices must be increased. But, increasing security implies a rising energy consumption. Otherwise, it is possible to integrate security solutions into small devices [8]. But, these are special solutions and are implemented in a device-specific manner. Thus, a sensor or actuator supplier will not invest in such solutions, because they are expensive and licensing costs may be higher than the price of the device.

At the Communication Layer *EnOcean* is used to collect all sensor data. EnOcean is a proprietary protocol, which we use in a pre-2012 version. Every security breach at the protocol layer will allow an attacker to tamper data or to influence actuators by providing wrong control data. EnOcean provides a developer kit, which allows encoding or decoding of data. In a normal setup the developer kit is used to program applications. However, it can be used by an attacker to listen to communication (e.g., eavesdropping), to send data, and to intercept messages. Replay attacks are also possible, since EnOcean supports only an easy mechanism for nonce generation. To increase the security, in 2012 a new protocol version [9] was introduced and the standard was adapted to [10]. This new protocol reduces replay attacks by inserting a rolling code into the telegram, but this is not enforced. Based on the protocol layer and the developer kit it is easy for an attacker to integrate self-crafted sensors or actuators into the system and manipulate data.

All collected data is transmitted from EnOcean, *Energy Production & Storage*, *Smart Meters* per office, and *Thermal Management Systems* to their *Wrapper* components and services by TCP/IP upon a non-secure IP layer. Thus, an attacker can use known IP exploits to attack the system. This enables eavesdropping, data tampering, and message manipulation. With known TCP/IP SYN flooding it is possible to generate DoS, which will block the *Building Energy Backbone* and all reliant services.

At the next level we use *Apache ActiveMQ* as a message broker for building the Building Energy Backbone or *Service Bus* for the device layer and the control layer. Every component or system user exchanges data through this message broker without authentication and authorization. This allows an attacker to attack components, influence user settings, and cause confusion. Besides, Apache ActiveMQ is also known to be prone to Cross Site Scripting (XSS)[1] and Cross Site Request Forgery (XSRF)[2]. XSS and XSRF are computer security vulnerabilities. XSS enables an attacker to inject client-side scripts to bypass access controls. XSRF forces an end user to execute unwanted actions.

The center of our ICT architecture, namely the Processing and Control Layer, resides above the Building Energy Backbone (Figure 1). It consists of a database, a user interface, and server applications, which use sensor information to calculate control commands. We use MySQL as a database. An attacker could use zero-day-exploits or specific attacks, e.g., Structured Query Language (SQL) injections or buffer overflows. SQL injections try to convince an application to run SQL code that was not intended. When a program is writing data to a buffer and overruns the buffer's boundary to write in adjacent memory, this is called a buffer overflow. After a successful attack it is possible to generate wrong data, leak information, or raise privileges.

An attacker could also compromise the User Interface (UI). This allows him to manipulate environment settings and to tamper control commands. For example the Stuxnet malware manipulated the UI without user's awareness. It attached additional control commands to the later deployed source code. In our system a malware could change settings silently before the data is transferred to *Server Applications*. These settings will be propagated to the Processing and Control Layer. Thus, it is possible to change our system rules, influence the sensor and actuator wrappers, and manipulate the remaining services.

On part of our server applications is the *RuleSystem*. Adding special crafted rules by an attacker can harm the system, such as executing wrong commands, changing privilege, or shutting down the system.

VI. COUNTERMEASURES

After the security audit of our Living Lab system we now describe the countermeasures. We improved our system security by closing the problematic gaps. Our counter measurements are closely related to our demonstrator, but are reported here to be used by others as a blueprint to provide security to their SOA-based prosumer systems.

[1]http://www.cvedetails.com/cve/CVE-2011-4905/
[2]http://fusesource.com/issues/browse/MB-670

A. *Countermeasures against external weak points*

Our *Two Way Meter* reads only the electricity consumption data given by the power meter. For data integrity we use a plausibility check. This analyzes the data whether it contains values in a given range. Besides that, we check timestamps to ensure that the given data is valid at the current time. As the meter is a legacy component we have to rely on the manufacturer's security mechanism.

We use the *Connector* as a security gateway. This component acts as a proxy for connections with the periphery of our system. It hinders attackers to get access to the internal system, because they must bypass the proxy first. The proxy prevents attackers also from collecting information about the system internals. Even a denial-of-service (DoS) attack affects only the proxy so that the internal system is operating normally.

Additionally, the proxy generates a token for every external client per session. This token is used as an authentication of an external user and allows an external client to receive system specific data. When the token is invalid or no token is presented we shut down the communication link immediately. We also use tokens to avoid cross site attacks. As a downside of proxy solution it is impossible to have end-to-end communication between an internal component and an external one.

Our proxy authorizes external clients with an access control system, which is based upon policies and user-specific trust levels. Authorized and trusted clients can access original system data. These clients can use the system data to collect data for statistical reasons or for billing purposes. On the other hand authorized, but untrusted, users are not allowed to access original data. In this case we provide semantically correct values by using format preserving encryption (FPE) [11]. The client will use the data and perform its calculation without noticing that the data only looks original. Since the client's data processing cannot distinguish original from masqueraded data, clients cannot derive their trust level from the observed data.

B. *Countermeasures against internal weak points*

In previous section we provided countermeasures against external security threats. Now, we introduce our solutions against internal security threats based on the analysis. We proceed from the lowest architecture level, the Device Layer, upwards.

At the Device Layer we use sensors and actuators, which have their identifiers burned into the ROM. This makes it infeasible to give these devices system-specific unique identifiers and change the identifiers in a random time period. In contrast, changing the identifiers would allow to recognize immediately when a device has been replaced or added to the system. However, since we built our system mainly from components of the shelf (COTS), we cannot apply such intrusive changes to the device.

In other words our devices are black boxes, which have to be used as they are provided. Thus, we decided to use wrappers to decouple devices from the system layer. These wrappers are used to translate device specific protocols to our common infrastructure protocols, e.g., IP. Moreover, the wrappers can apply sanity and consistency checks to the data provided by the black box devices, e.g., data is validated against a defined range of expected values. Based on this behavior model and the validation results, the system decides whether a device operates normally or data must be discarded until the problem has been fixed.

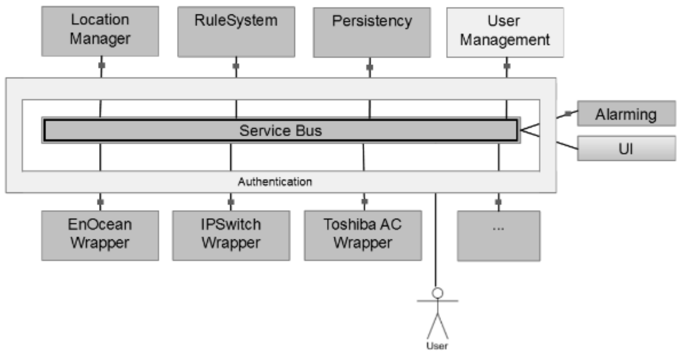

Fig. 3. Enhanced *Living Lab* demonstrator architecture with components [15]

At the Communication layer we use the specific protocol security of EnOcean. For the IP communication we migrated from the plain version to the IPsec one with ESP mode [12]. This migration affected our Service Bus, which uses Java Message Service. Here, we had to change our configuration and implementation to send all messages over IPsec. With this step towards an IPsec system a key management system is needed. The introduction of a key management system will be done in a future version of our system. To avoid TCP/IP SYN floods and other IP or TCP/IP attacks we adopt the solutions given in [13].

Currently, our Service Bus allows connections of unauthenticated components or users and it does not prevent unauthorized data access. Thus, we use the Pluggable Authentication Module (PAM) [14] framework for authentication. PAM is a mechanism to integrate low-level authentication schemes into applications. The authentication at the application layer can be written independent of the underlying schemes. We implemented the PAM concept by using the Java Authentication and Authorization Service (JAAS).

After an integration of authenticated components we also have to guarantee that system privileges are correctly evaluated. System privileges are realized by a role-based access control as described in [15]. The evaluation of privileges is done by different authorization services. We use the *Apache CXF* framework, because it supports interceptors.

Interceptors [16] are similar to filters and can modify the communication between the server and a client without their notice. Interceptors can be activated on demand and can then augment or change messages if necessary.

Figure 3 shows our improved architecture. The interceptors are visualized as black boxes in front of each component and the Service Bus is guarded by our authentication and authorization solution.

For the security countermeasures of the Processing and

Control Layer we start with the RuleSystem. Here, we only deal with confidentiality. Authentication is already done by the Service Bus, while integrity will be done in the next break. Internally, rules are clustered into three different policy levels: high, medium, and low confidentiality. All basic system rules are stored in the high security level. These rules are mainly defined by system manufacturers and provide essential functionality. It is only possible to replace or add such rules with specific privileges, e.g., administrative authorization is required. The medium security level only contains certified rules, while in the low security level every user, service, or hardware component is allowed to add or change rules.

Based on that clustering, it is possible to prioritize rules and their activation. Our rule system checks that no rules violate any rule in a higher security level. In addition, high and medium level rules will not contradict each other at the same level. Low confidentiality rules, which are provided by arbitrary users, can be contradictory to each other and even higher level rules. Technically, before changing the active rule set of the rule engine, the new set is checked for non-deterministic behavior with respect to the actuator by the SAT-solver YICES [17]. When non-deterministic behavior is found, the activation is prohibited.

Additionally, we introduce an anomaly-based intrusion detection system (IDS). This is needed for an active scanning of message exchange. Therefore, we can identify misbehavior, e.g., in case an attacker tries to influence the system with manipulated traffic and services data. Our IDS uses a system model, which gets information from all connected agents. The agents collect the information at specific interfaces of our wrappers, the Connector, and the RuleSystem.

As a summary the authors are confident that the counter measurements presented here have improved the security of our Living Lab. However, as always in security we cannot prove that there are no other attack points.

VII. Conclusion

In this paper we analyzed our prosumer system, the *Living Lab*, where we used our defined attack model. In order to define the attack model we took a closer look at common attack motivations. We used the results of this analysis to identify the weak points of our prosumer system. Finally, counter measurements have been identified and concrete implementations and extensions to the system architecture have been proposed. Hereby, we have demonstrated the feasibility of building and enforcing practical security for prosumer systems.

Acknowledgment

The authors would like to thank the EIT ICT Labs for the support to build up the demonstrator.

References

[1] D. Koss, D. Bytschkow, P. Gupta, B. Schatz, F. Sellmayr, and S. Bauereiss, "Establishing a smart grid node architecture and demonstrator in an office environment using the SOA approach," in *Software Engineering for the Smart Grid (SE4SG), 2012 International Workshop on*, june 2012, pp. 8 –14.

[2] J. D. Howard and T. A. Longstaff, "A Common Language for Computer Security Incidents," Sandia National Laboratories, Albuquerque, New Mexico 87185 and Livermore, California 94550, Samdia Report SAND98-8667, October 1998.

[3] R. J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 2nd ed., ser. Wiley computer publishing. Wiley, 2008.

[4] J. Chinnow, K. Bsufka, A. Schmidt, R. Bye, A. Camtepe, and S. Albayrak, "A simulation framework for smart meter security evaluation," in *Smart Measurements for Future Grids (SMFG), 2011 IEEE International Conference on*, nov. 2011, pp. 1 –9.

[5] N. Mead, E. Hough, and T. S. II, "Security Quality Requirements Engineering," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania,, Technical Report CMU/SEI-2005-TR-009, 2005. [Online]. Available: http://www.sei.cmu.edu/library/abstracts/reports/05tr009.cfm

[6] B. Schneier, "Secrets and lies - digital security in a networked world: with new information about post-9/11 security." 2004, pp. I–XIII, 1–414.

[7] ——, "Attack trees: Modeling security threats," *Dr. Dobb's journal*, December 1999.

[8] M. Vogt, A. Poschmann, and C. Paar, "Cryptography is feasible on 4-bit microcontrollers - a proof of concept," in *International IEEE Conference on RFID*, Orlando, USA, 4 2009, pp. 267–274.

[9] E. GmbH, *Security of EnOcean Radio Networks*, v 1.2 ed., Kolpingring 18a, 82041 Oberhaching, Germany, Juli 2012.

[10] *ISO/IEC 14543-3-10:2012: Information technology – Home Electronic Systems (HES) – Part 3-10: Wireless Short-Packet (WSP) protocol optimized for energy harvesting – Architecture and lower layer protocols*, International Organization for Standardization Std., 2012.

[11] M. Bellare, T. Ristenpart, P. Rogaway, and T. Stegers, "Format-Preserving Encryption," *IACR Cryptology ePrint Archive*, vol. 2009, p. 251, 2009.

[12] S. E. Frankel, K. Kent, R. Lewkowski, A. D. Orebaugh, R. W. Ritchey, and S. R. Sharma, "Sp 800-77. Guide to IPsec VPNs," National Institute of Standards and Technology, Gaithersburg, MD, United States, Tech. Rep., 2005.

[13] M. Atighetchi, P. Pal, F. Webber, R. Schantz, C. Jones, and J. Loyall, "Adaptive Cyberdefense for Survival and Intrusion Tolerance," *IEEE Internet Computing*, vol. 8, no. 6, pp. 25–33, Nov. 2004. [Online]. Available: http://dx.doi.org/10.1109/MIC.2004.54

[14] V. Samar, "Unified login with pluggable authentication modules (PAM)," in *Proceedings of the 3rd ACM conference on Computer and communications security*, ser. CCS '96. New York, NY, USA: ACM, 1996, pp. 1–10. [Online]. Available: http://doi.acm.org/10.1145/238168.238177

[15] L. Cito, "Design and Implementation of a Right and Role Management System for the fortiss Smart Energy Living Lab," 2012, bachelor thesis.

[16] J. Ramachandran, *Designing Security Architecture Solutions*. New York, NY, USA: John Wiley & Sons, Inc., 2002.

[17] B. Dutertre and L. de Moura, "The yices smt solver," Computer Science Laboratory, SRI International, Tech. Rep.