Interactive Rule Learning for Access Control: Concepts and Design

Matthias Beckerle, Leonardo A. Martucci, Sebastian Ries, Max Mühlhäuser

Telecooperation Group (TK), Technische Universität Darmstadt
DE-64293 Darmstadt, Germany
{beckerle, leonardo, ries, max}@tk.informatik.tu-darmstadt.de

Abstract—**Nowadays the majority of users are unable to properly configure security mechanisms mostly because they are not usable for them. To reach the goal of having usable security mechanisms, the best solution is to minimize the amount of user interactions and simplify configuration tasks. Automation is a proper solution for minimizing the amount of user interaction. Fully automated security systems are possible for most security objectives, with the exception of the access control policy generation. Fully automated access control policy generation is currently not possible because individual preferences must be taken into account and, thus, requires user interaction. To address this problem we propose a mechanism that assists users to generate proper access control rule sets that reflect their individual preferences. We name this mechanism Interactive Rule Learning for Access Control (IRL). IRL is designed to generate concise rule sets for Attribute-Based Access Control (ABAC). The resulting approach leads to adaptive access control rule sets that can be used for so called smart products. Therefore, we first describe the requirements and metrics for usable access control rule sets for smart products. Moreover, we present the design of a security component which implements, among other security functionalities, our proposed IRL on ABAC. This design is currently being implemented as part of the ICT 7th Framework Programme SmartProducts of the European Commission.**

Keywords**-*****adaptivity*****,** ***usability*****,** ***access control*****,** ***rule learning*****.**

I. Introduction

Smart products are a new class of upcoming devices that is currently been developed to bridge the gap between the real and the virtual world. They provide a natural and purposeful product-to-human interaction and context-aware adaptivity. Smart products need knowledge about the application, the environment that they are immersed in, and confidential user data, such as user's preferences and behavioral information, to fulfill their tasks. Moreover, smart products often need to exchange private / confidential information between each other to complete collaborative tasks that require data from multiple sources, such as booking flight tickets or hotel rooms. Smart products can be part of highly dynamic environments.

However, the amount of possible security breaches is proportional with the sheer number and variety of smart products. Equally, the variety of devices with different user interfaces also increase the complexity of administrative tasks for the end-users. Therefore, one of the main challenges of IT-security regarding smart products is the design of mechanisms that combine a customizable level of security and usability [1]–[3].

Current IT-security solutions tend to overstrain non-expert users. In home and enterprise environments, users are frequently forced to choose passwords for local and remote authentication and also define rules for access control, e.g., file sharing access rights. However, the imposition of such security features often lead to insecure or unpractical measures, such as written passwords and access control rules that are often too general. In addition, users tend to deactivate security mechanisms or render them useless by: not changing default passwords or leaving them blank; granting access to everyone; or turning off basic security mechanisms. This kind of behavior is very common nowadays, especially regarding login passwords, browser cookies, virus scanners, and file access controls [4].

The administration of secure features in computational systems by non-expert end-users is already a challenge. Such a fact can be easily shown by the massive number of computers that are part of bot nets [5], which is, in most of cases, caused by inability of such users to keep their systems up-to-date or to change default settings. Smart products add more complexity to such scenarios by increasing the administrative burden to the end-users.

In this paper, the usability aspects of security solutions are first analyzed. This initial analysis is used to identify usability gaps in basic security mechanisms that can be applied for smart products. It shows that there are already sufficient solutions that can applied for confidentiality, integrity, and authentication services, but it also concludes that no appropriate access control solutions exist nowadays that take user preferences into account.

We define security and usability requirements for access control rule sets that can be implemented in smart products. We then propose a mechanism that allows more user-friendly access control rule generation and provide a design for a security architecture for implementing it. The architecture is presented as a component that is integrated to a larger software platform being implemented as part of the ICT 7th Framework Programme SmartProducts of the European Commission.

This paper is organized as follows. In Section II, some key terms are defined. Section III brief outlines the state-of-the-art solutions for maintaining confidentiality, integrity, authentication, and authorization in highly dynamic environments. In particular, authorization and access control mechanisms are analyzed in Section III-E. An access control model suitable for smart products is presented in Section IV and security and usability requirements for access control rule set are introduced in Section V. In Section VI, we present a solution that helps users to generate proper access control rule sets using a combination of automated rule learning and user interaction. The related work is shown in Section VII. The design of a security architecture for smart products and its underlaying building blocks are presented in Sections VIII, IX and X. Finally, the concluding remarks are presented in Section XI.

II. Definition of Security Terms

In this section, we define the some key security terms that are going to be use throughout this paper: reliability, usability, confidentiality, integrity, authenticity and authorization.

- **Reliability**: in this paper we define reliable security as a set of security mechanisms that is able to fulfill the security expectations of an end-user regarding their security requirements.
- **Usability**: in this paper usability means that security mechanisms demand minimum user interference to be deployed. A smart product should stay as usable as it would be without security mechanisms. Thus, the introduction of security should be preferably automated.
- **Confidentiality**: means that the assets of a computing system are accessible only by authorized parties. Confidentiality is usually implemented using cryptographic algorithms.
- **Integrity**: means that assets can be modified only by authorized parties or only in authorized ways. Integrity is mostly implemented using one-way functions in combination with cryptographic algorithms.
- **Authenticity**: means that an entity can prove who or what they claim to be. Authentication services are usually implemented by a proof of knowledge, a proof of ownership, or a proof of biometric trait.
- **Authorization**: means that policies are used and enforced to specify access rights. Authorization is implemented through access rules that are used by access control mechanisms to determine if an entity is allowed to access information or not.

The aforementioned terms reliability and usability are often seen as contradicting goals, especially regarding access control rules. Such contradiction is usually resulting from the huge amount of rules that are required to secure a system, which makes them unintelligible for end-users. Usability in most cases is simply neglected, what can result in insecure systems in the long-term since users tend to turn such security features off or use them in improper ways, as mentioned in Section I. These dependencies are shown in Figure 1.

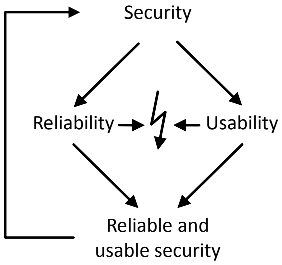

Figure 1. Dependencies for reliable and usable security

III. Security Services for Smart Products

To achieve reliable and usable security, an analysis of existing security services in the context of smart products and highly dynamic environments is needed first. This section presents such an analysis. In such a context, we show that confidentiality, integrity and authenticity can be automated quite well, but authorization cannot. Confidentiality is presented in Section III-A, integrity in Section III-B, authenticity in Section III-C, and authorization in Section III-D.

A. Confidentiality

For a reliable secure system it is important to secure not only the access to the data, but also to secure the data itself, whereas stored or in transit. Confidentiality can be achieved using encryption to protect data.

There are symmetric, such as AES, and asymmetric encryption mechanisms like RSA. Symmetric key encryption demand the distribution of cryptographic keys among participating devices. Asymmetric key encryption performs worse than symmetric key encryption. Hence, large chunks of data are rarely encrypted using asymmetric keys, but only selected data, such as symmetric keys. In smart products, the process of symmetric key distribution is a potential challenge because if a unique key is demanded for every pair of communicating entities, the number of required keys equals $\binom{n}{2}$, where n is the total number of communicating devices. Nonetheless, it is feasible to embed public-private key pairs into them, which would reduce the number of total number of keys to $2n$. Such an approach is sufficient in principle and implements confidentiality into high dynamic environments using existing and standard cryptographic systems.

B. Integrity

Integrity has to assure that any unauthorized change of data is recognized. Data integrity is usually accomplished using one-way hash functions and public key encryption or with just symmetric keys. Message Authentication Codes (MAC) [6] are implemented using symmetric keys and digital signatures with public-private key pairs. Since such cryptographic tools are expected to be embedded into smart products (as seen in Section III-A), there are going to be enough cryptographic tools available for securing data integrity.

C. Authenticity

Authentication is required to obtain a proof of correctness over an identity claim. In smart product scenarios there are basically three types of authentication: device–to–device, device–to–user, and user–to–user. There are sufficient mechanisms based on digital certificates that can carry out device–to–device authentication automatically. Device–to–user and user–to–user authentication can also be realized using proofs of knowledge, biometric traits or digital tokens together with public-key encryption. In such a case, after users authenticate themselves to smart products, such devices might be used to automatize other authentication procedures between users and other devices.

D. Authorization

Authorization is needed to specify access rights and enforce them. It is implemented through access rules, and the collection of such rules is referred to as a rule set. There are mechanisms that allows fully automated generation of rule sets for smart products. Such approaches, however, disregard adaptivity to the end-user. The general problem is resulting from the diversity of user preferences, so more information regarding the users is required. Authorization problems regarding adaptivity and user in smart products are discussed in Section III-E, where the existing access control models are outlined and evaluated regarding their suitability to smart product scenarios.

E. Access Control

This section provides an overview of different access control models and provides an evaluation of such models regarding their suitability to smart product scenarios. In this section, we describe the following access control (AC) models: Blacklists, Mandatory AC (MAC), Discretionary AC (DAC), Role-Based AC (RBAC), and Attribute-Based AC (ABAC). This section concludes with a set of recommendations for an AC models suitable for smart product scenarios. It concludes that ABAC models together with Blacklists is the most suitable solution for such scenarios.

The role of AC mechanisms, which are implemented after AC models, is to ensure that only authorized entities are able to access the information and functions of a computer system (principle of authorization) [7].

1) Blacklist: A Blacklist AC is a very simple AC that blocks all requests from entities that are included in a Blacklist. It is used to thwart known or recurrent attackers. Blacklists have to be configured manually or, sometimes, they can be updated automatically according to predefined rules, e.g., multiple unauthorized requests, or a series of failed authentication procedures. Blacklists usually outperform other AC mechanisms because their complexity class is lower than those, and its performance can be $\mathcal{O}(1)$ with a very small constant factor for the blacklist lookup. Blacklists are a rather simple to use AC, but also rather inflexible, since there no conditional access policies can be defined.

2) MAC/DAC: MAC and DAC are two early AC models [8]. MAC and DAC can be seen as complementary approaches, but both link access rights directly to the related entities.

In MAC, a central administrator controls the access rights of each entity of the system. No other entity is able to change the access rights. In such a context, MultiLevel Security (MLS) (such as Bell-La Padula [9]) is an often used approach. In MLS, each entity or object of the system has a security level given by a central authority. Each entity is only able to access other entities or objects that have the same or a lower security levels. Mandatory Integrity Control (MIC) is a similar approach and is used in Microsoft Windows Vista (and later). Processes can only write or delete other objects with an security level lower or equal to their own.

DAC differs from these approaches as each entity can hand its rights over to other entities. That way, users are able to share objects among each other. DAC is used in UNIX and Windows-based systems for sharing data and resources.

3) RBAC: RBAC [10] introduced a new way by setting roles between the entity and the related rights. That way, each entity can have several roles and each role can be held by multiple entities. For administrative purposes, roles are established first, and afterwards they are assigned to entities. Since roles usually rarely change, this reduces the complexity for administrating RBAC significantly after the first setup. If only those entities change that inherit a role, this can be simply addressed by adding or deleting entities (in form of the name or a unique identifier) that are associated with the regarding role. Roles can change dynamically and in that way the user might gain and lose roles automatically when doing special tasks.

4) ABAC: One of the newest models is ABAC [11]. ABAC uses attributes instead of roles to link rights to entities. This procedure allows the use of dynamic conditions encoded in attributes, such as the location of an entity, to decide whether to grant access or not. Since the role as well as the security level of an entity can be seen as an attribute, it is possible to integrate concepts known from other AC models like DAC or RBAC.

5) Hybrid approaches: In reality, the distinction between different AC models is not as strict as shown in this section. There are hybrid models like the Location-Aware Role-Based Access Control (LRBAC) [12], which allows the use of a geographical location as a “role”. It is often possible to derive a less complex AC model from a more complex one, e.g., it is possible to create an MAC mechanism from

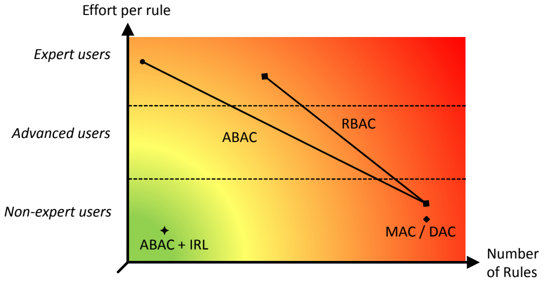

Figure 2. Theoretical comparison of different AC models.

an ABAC model.

IV. Access Control for Smart Products

Smart products are user adaptive devices which require AC mechanisms with maximum flexibility since they are related to the everyday life of a heterogeneous set of end-users. Smart products need to maintain user profiles that have attributes and values about users, such as preferences to fulfill their tasks. ABAC models are an evident candidate for building up AC mechanisms for smart products because they provide maximum flexibility in comparison to the aforementioned AC models.

ABAC models are, however, more complex than the other models listed in this section. Such complexity results in a larger consumption of computational resources than simpler approaches. Thus, to reduce to costs of AC operations a Blacklist AC mechanism can be executed before the ABAC mechanism. The Blacklist filters out known misbehaving entities, and their requests do not reach the ABAC mechanism. For instance, after an entity, that was not blacklisted at first, has multiple identical requests denied by the ABAC mechanism, such an entity can be temporarily or permanently added to the blacklist.

AC mechanisms like ABAC are dynamic and flexible. However, they are also hard to configure in the right way. While MAC and DAC have only one way to link access rights to the user, RBAC and, especially, ABAC allows for different ways of connecting access rights to entities through indirect mapping. This flexibility enables very compact and meaningful policy sets. However, if not correctly used, it can lead to an heterogeneous and incomprehensible set of rules. This problem is very likely to occur in case of inexperienced users. This is an important challenge that is addressed with Interactive Rule Learning in Section VI.

The relation between flexibility of an AC mechanisms and the usability is shown in Figure 2. This figure shows that MAC / DAC can be used by non-expert users but the number of needed rules for non-trivial scenarios is extremely high. The figure also illustrates that RBAC and ABAC can have very short rule sets, however, only expert users might be able to do so (since it is difficult to manually define a minimal rule set for a complex scenario). If more rules are used in RBAC and ABAC, it is possible to emulate MAC / DAC mechanisms with the difference that always a role or an attribute is in between entities and their related access rights. Finally, the figure shows that ABAC plus Interactive Rule Learning can be used to create reduced rule sets even by non-expert users.

After defining a suitable AC model for smart products it is still fundamental to define how the rules for such AC model are generated. Such rule generation should consider a set of requirements that are discussed in the next section.

V. Requirements for AC Rule Sets

In this section, we define the requirements for AC rule sets for smart product scenarios taking into account both security and usability constraints. Not all rules presented in this section are orthogonal, thus conflicts do exist. Such conflicts are detailed and explained in the end of this section.

A. Security Constraints

The security constraints for building up AC rule sets are regarding specific or permissive rules and also the meaning of such rules. Each requirement is assigned a letter S followed by a number.

- $S1$: specific (permissive) rules. Access rules have to be specific enough to leave no opening for intruders. Rules like “everyone is allowed to do everything” render AC mechanisms useless in practice.
- $S2$: meaningful rules. Access rules have to reflect the expectations of the smart product owner. Rules like “every employee of the university is allowed to use the printer” have a better semantic meaning than a similar rule stating that “every one with glasses is allowed to use the printer”, even if every employee of the university wears glasses.

B. Usability Constraints

The usability constraints for building up AC rule sets are regarding the existence of redundant rules, their consistency and understandability, and also related to the total number of rules. Each usability requirement is assigned a letter U followed by a number.

- $U1$: no redundant rules. Rules or set of rules that are fully covered by other rules or set of rules can be deleted without changing the behavior of the AC mechanism. Thus, if a rule set A is a subset of a rule set B, then rule set A can be deleted. Redundant rules only increases the complexity of a rule set without adding any security features and make such sets more confusing for the end user.
- $U2$: consistent rules. Consistent rules mean that two or more different rules must not be contradictory. Contradictory rules could lead to unpredictable access

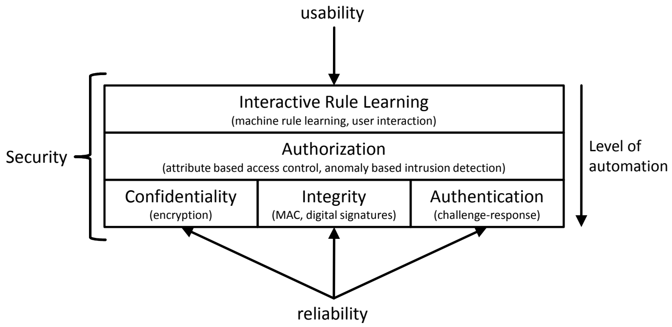
usability
Interactive Rule Learning
(machine rule learning, user interaction)
Security
Authorization
(attribute based access control, anomaly based intrusion detection)
Level of automation
Confidentiality
(encryption)
Integrity
(MAC, digital signatures)
Authentication
(challenge-response)
reliability

Figure 3. Reliable and Usable CIA + Authorization

decisions or worsen the usability by unnecessarily increasing the complexity of the rule set.

- *U*3: general, understandable and manageable rule sets. AC rules need to be general enough for users to understand and manage.
- *U*4: minimum number of rules. The number of rules that describes the scenario should be minimal to make the rule set understandable and manageable.

The use of general rules in requirement *U*3 contradicts the requirement *S*1 regarding specific rules. Thus, the best compromise between specific and general rules need to be reached. The best compromise is, however, connected to the users preferences and it is, therefore, individual.

Rule *U*2 is not only a usability requirement, since it can also impact the security level obtained by the AC mechanism. An inconsistent rule set can lead to a non-expected behavior that can compromise the security of the smart product.

In the next section, we develop a rule generation procedure that takes the aforementioned requirements into account. Such procedure combines automatic rule generation with user interaction.

VI. Rule Generation

Nowadays, the common procedure for rule generation is to do it manually. Therefore, the requirements listed in Section V need to be considered by the owner of the smart product. The manually generation of rules by inexperienced users will likely result in misconfigured access rule sets (or the manual deactivation of security mechanisms), which eventually end up into security vulnerabilities. Therefore, the rule generation process should be automated as much as possible. Learning algorithms, from the Artificial Intelligence research field, are able to accomplish this goal [13].

A. Automatic Rule Learning

Extracting knowledge out of data by using a rule-learning algorithm is a well-known topic. However, for defining good access rules, a fully automated rule generation is unfortunately not worth most of the time. It is very difficult to determine automatically what kind of information needs to be protected. The whereabouts of a person, for instance. Taxi drivers may have their geographical position public available, but for lawyers or doctors on their way to clients or patients must keep their location information strictly private.

It could be possible to decide which information should be public and private by analyzing the user profile. Thus, automatic rule set generation is possible, but it is expected that errors would also be a commonplace. However, if related information for automatic rule generation is missing, automatic processes are not possible. Hence, the smart product owners have to decide by their own regarding the access rules.

Therefore, a proper solution is to use automatic rule generation to create an initial rule set that is later presented to the user. Such a solution is presented in Section VI-B.

B. Interactive Rule Learning

Learning algorithms can generate a set of rule sets and present them to users that decide which specific rule set suits best to their context. A rule learner can be used to analyze the set of access rules of a smart product regarding the actual behavior of entities [14].

Such an analysis disclose whether rules are shadowed, redundant, or correlative, and which exceptions exist following the definition and classification presented in [15]. Furthermore, in interaction with users, the number of rules is minimized by analysing, pruning, and rebuilding the set of access rules. This procedure is called Interactive Rule Learning (IRL) [16].

Combined with the ABAC, the IRL helps the user to build a secure and usable set of access rules. The expected outcome of ABAC+IRL is shown in Figure 3. This concept represents an important step towards usable security.

An automated rule learning algorithm can fulfill the following requirements: *S*1, *U*1, *U*2 and *U*4. Users have to verify the compliance of requirement *S*2, since it depends on the context and also on the smart product owner preferences. To satisfy requirement *U*3, regarding general rules, interaction between the smart product owner and the rule leaner is required.

VII. Related Work

Over the years, a variety of learning algorithms have been developed that try to imitate natural learning or use a more technical approach as a starting point. Some approaches try to reproduce the functioning of a brain at the level of neurons [17], [18]. Other mechanisms, such as support vector machines, are based on a more abstract mathematical concept by finding an optimal border between positive and negative examples (like access and deny for an access request) by maximizing the distance between them [19]. Existing algorithms further differ with respect to their applicability, speed, and accuracy [20], [21].

Rule learners use a very intuitive approach in relation to the aforementioned algorithms. They try to find causalities in recorded databases and express them with simple rules. For example, in a database that describes the attributes of different animals like ravens, sparrows and pigs such a rule could be as follows: “If an animal can fly and has feathers, it is a bird”. This approach has the particular advantage of being relatively easy to understand for humans as opposed to the classification of a support vector machine, for instance. This is both a psychological and a practical advantage. From a psychological perspective people tend to accept something more likely if they are able to understand it. From a practical point of view, potential errors can be more easily detected and extended [16].

VIII. Designing Security for Smart Products

Usable access control mechanism for smart products is currently being implemented as a component of the SmartProducts software platform [22] being implemented as part of the SmartProducts project funded by the European Commission’s 7^{th} Framework Programme. In this section we describe the design overview of the Access Manager component in the SmartProducts software platform. The objective of this platform is to provide an open framework for developers to design hardware and implement applications for smart products. The Access Manager component is mainly responsible for the authentication, access control and security administration in smart products. Figure 4 depicts the architecture of the Access Manager component. This diagram and the following diagrams in this paper are presented using the FMC notation [23].

The Access Manager has interfaces to the Communication Middleware, which handles all communications outside the device, and to the Ubiquitous Data Store, which implements an interface to one or more databases. The Access Manager has three subcomponents: the Authentication, the Access Handling, and the Security Administration. The functionality of the aforementioned subcomponents are summarized next:

- The *Authentication* subcomponent handles multiple authentication mechanisms, such as biometric data (BD).It is responsible for authenticating entities, such as users and devices. The authentication component is out of the scope of this paper and it is not going to be further detailed and it is only mentioned for the sake of completeness. Anyhow, it is a fundamental building block of the security architecture for smart products.
- The *Access Handling* (AH) manages the blacklist and the ABAC as deducted in Sections III-E and IV. The AH is described in details in section IX.
- The *Security Administration* (SA) is the component where the Interactive Rule Learning (see Section VI-A) takes place. The SA is described in details in section X.

IX. Access Handling

The Access Handling assures that only authorized entities are able to access the proactive knowledge (PaK), which is basically a secure distributed database for RDF (Resource Description Framework) data and key-value pairs[1]. It filters every request through a set of rules and forwards only those requests that are legitimated. The Access Handling is pictured in Figure 5. This section is divided into two parts: the first part describes the access handling components, and an use case is illustrated in the second part.

The Access Handling is composed by the Access Handler (AH), the Blacklist Handler (BH), the Access Control (AC), and the Intrusion Detection System (IDS), which are detailed next.

A. Access Handling Components

1) Access Handler: The AH is the interface for access requests. It forwards requests to the BH and informs the requesting entity about the outcome of the request. If an access request was authorized by the security system it will be forwarded to the Ubiquitous Data Store. If it is necessary to send data back to the requesting entity to fulfill the request, the data goes through the AH.

2) Blacklist Handler: The BH is the first of the three rule-based access control mechanisms. It blocks every request from entities which are listed in the General Blacklist database. If the requesting entity is not blocked the request is forwarded to AC for further examination.

3) Access Control: The AC checks if a request corresponds with the access rights of the requesting entity. All access rights are read from the AC Rules database. If an access is refused, the ID of the requesting entity will likely be added to the General Blacklist database for a period of time. If an entity often tries to access a resource it has no authorization for it will be added to the General Blacklist database. In all cases a blocked request will be sent to the AH to inform the requesting entity that it was blocked. If a rule exists that allows the requesting entity to access the requested resource the request is forwarded to the IDS.

4) IDS: The IDS verifies that an access request does not deviate from the expected behavior of the requesting entity. The expected behavior is stored in the Behavior Whitelist database. If an access request is determined as abnormal the IDS asks the IDS of other smart products around for help. The IDSs around combine their knowledge about the requesting entity to determine if the entity is the one it claims to be. The outcome will be used to update the database for better future results and is also forwarded to the AH. The IDS should ideally be able to exchange information with IDSs on other smart products in order to cooperatively detect intrusions. The design and implementation of the IDS component is out of the scope of this paper.

[1] More information regarding PaK is available on the SmartProducts project website: www.smartproducts-project.eu.

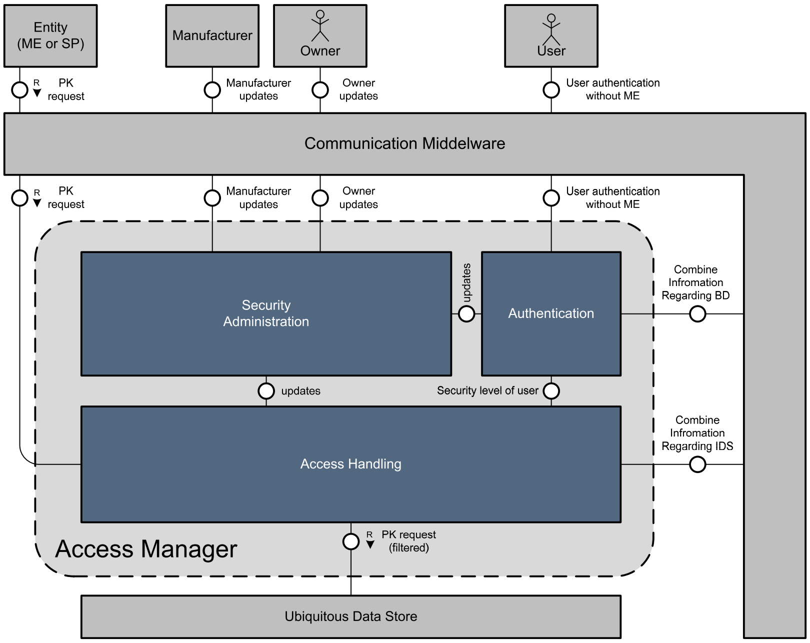

Figure 4. Access Manager architecture.

B. Use Case

In this use case, an user request information from the Ubiquitous Data Store. To decide whether a data access may take place or not, the respective request must pass through all mechanisms of the Access Handling. Only when all instances approve the right to access the data, the request will be forwarded from the AH to the Ubiquitous Data Store. This use case is depicted in Figure 6.

X. Security Administration

The Security Administration contains the Rule Handler (RH) – a bidirectional interface for the owner, the manufacturer and the Access Handling of the smart product to update the access rules. The RH maintains three Databases for the different Access Handling mechanisms. Every smart product in the same Trusted Network [24] of the owner has the same owner specific access rules for redundancy and against easy manipulations. To help the user define suitable rules a new research topic called Interactive Rule Learning will be investigated. The Security Administration Module is depicted in Figure 7. This section is divided into two parts: the first part describes the Security Administration components, and an use case is presented in the second part.

A. Security Administration Components

The Security Administration is composed by the Rule Handler (RH), the set of access control rules, the blacklist and the whitelist. The RH exchanges information with the Minimum Entity (ME). A description of the RH and the ME are provided below.

1) Rule Handler: The RH manages the three databases General Blacklist, AC Rules, and Behavior Whitelist. The General Blacklist contains the identities of blocked entities used by the BH. Moreover, there are two different databases for AC rules: The first database contains access rules defined by the user, the second database consists of access rules of the manufacturer, which are required for maintaining the smart product (e.g., firmware updates). The Behavior Whitelist has rules that describe the normal behavior of entities interacting with the smart product and is needed for the IDS.

The RH communicates to the Minimal Entity (ME) or equivalent device to support the user managing the different databases. This is done with the support of Interactive Rule-Learning (see [24]). The manufacturer of the smart product is only able to update the manufacturer AC Rules database.

2) ME: The ME is a device that represents the owner in the digital world. It is used to easy authenticate the owner and as a user interface for configuring the RH. Alternatively the functionality can be integrated in a smart product. MEs are described in detail in [24].

B. Use Case

In our scenario, we consider a family of four. Alice (*A*), Bob (*B*), and their children Charlie (*C*), a 17-year old, and Denise (*D*), an 8-year old. The set with elements $\{A, B, C, D\}$ is the family, and the subset with elements $\{A, B\}$ are the parents. In the family's kitchen there are 3 new smart products: a smart coffee machine (*X*), a smart blender (*Y*), and a smart oven (*Z*).

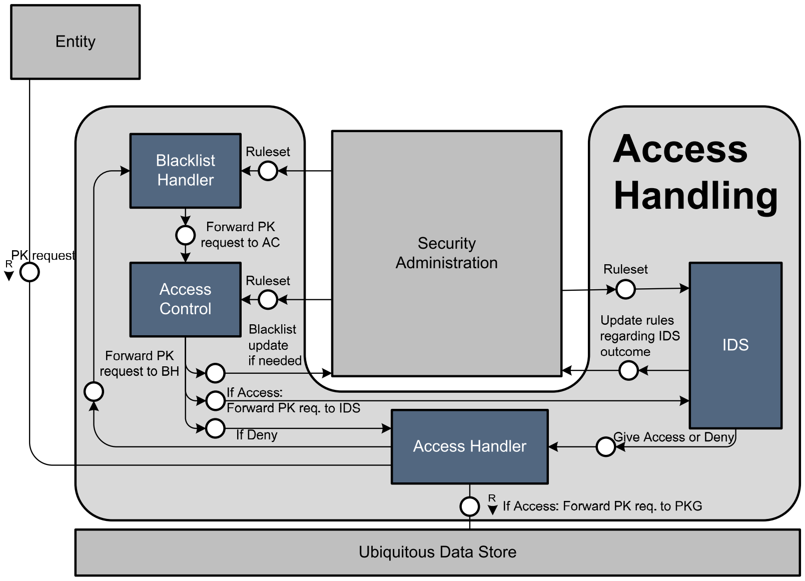

Figure 5. Access Handling Component Overview.

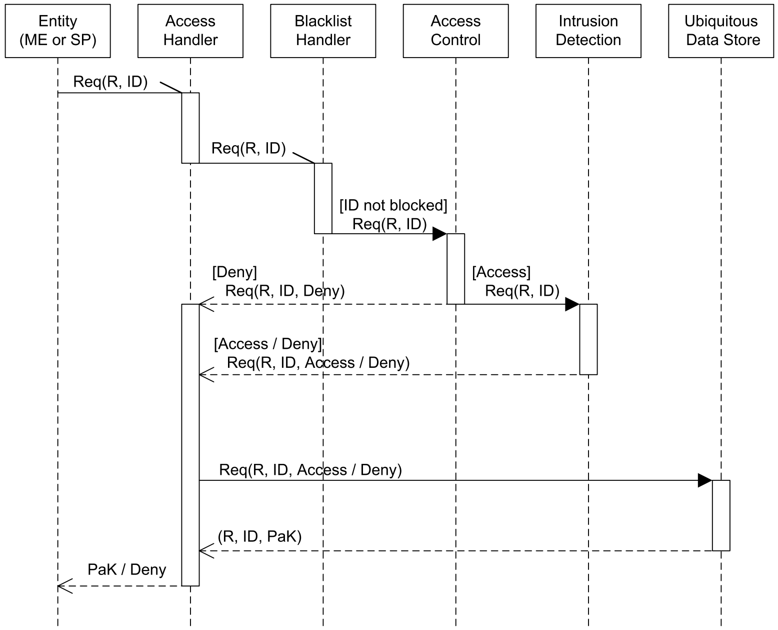

Figure 6. Access Handling Use Case.

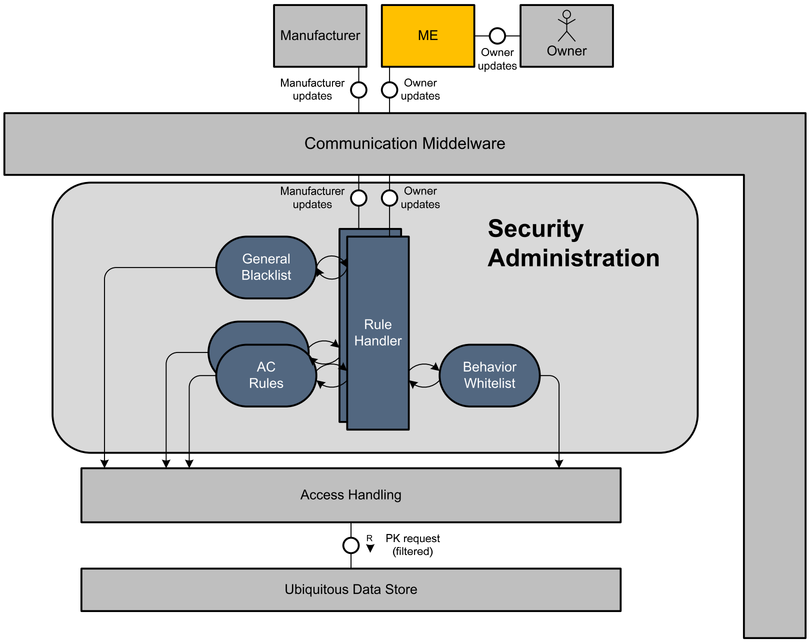

Figure 7. Security Administration Module.

We assume that newly bought devices come with a default set of access rules, which are defined by the smart product manufacturers. Since the manufactures cannot predict in which way smart products are going to be used, the factory settings for the access control rules are basically general. They follow the usage rules of similar non-smart products, i.e., everyone that physically interacts with a device is allowed to use up to its full-functionality. For instance, everyone locally interacting with the coffee machine is allowed to brew coffee.

The full control of a smart product is given to the one who first activates it. A smart product might be remotely controlled by its users (through smart devices) after it has been integrated into the home environment. *A* wants to configure and generates access rules for the 3 newly bought smart products (*X*, *Y*, *Z*), so that her family can best profit from them. Three classes of access rights are preloaded in smart devices (those classes can later be reconfigured or changed):

1) *Full access*: the right to locally or remotely access a smart product and to manage its access rights.
2) *Remote and local access*: the right to locally and remotely access a smart product.
3) *Local access*: the right to locally access the smart products.

A wants to grant *B* with full control over all the smart products. *C* shall get access to the full functionality (locally and remotely), but shall not have administrative rights over the smart products. *D* shall not have any access to the devices, even by local interaction. Since the family often have guests, *A* wants them to be able to locally interact with the smart products, just as in a non-smart kitchen. The initial manually generated rule set is:

```
Rule Set 1
 1: If (owner) -> full access
 2: If (any)   -> local access
 3: If (A)     -> full access
 4: If (B)     -> full access
 5: If (C)     -> remote and local access
 6: If (D)     -> no access to X
 7: If (D)     -> no access to Y
 8: If (A)     -> no access to Z
 9: If (guest) -> local access
```

There are a few mistakes in *A*'s manually generated rule set. They are:

- The first two rules are residues from the preloaded factory default rule set. The fact that *A* ignored them leads to two implications regarding requirements *U*1 and *U*2. Rule 2 is a superset of rule 9, and it also contradicts rules 6, 7, and 8. Moreover, since there are redundant rules, their number is surely not minimum, which contradicts *U*4.
- Rule 9 is misconfigured since it does not reflect *A*'s expectation. Instead of denying *D*, she denied herself to access *Z*. It contradicts requirements *S*2 and *U*2.
- The rules were generated taking into account specific family members instead of more general attributes, such as age. The use of attributes for generating small and understandable rule sets is recommended and one of the reasons why ABAC is better suited for smart products, as mentioned in Section III-E. Therefore, there is a contradiction with *U*3.

Table I
USABILITY OVERVIEW TO CIA AND AUTHORIZATION.

	Confidentiality	**Integrity**	**Authenticity**	**Authorization**
Usability	Yes, transparent	Yes, transparent	Yes, transparent	Partially, fully automation not possible
Adequate Method	Encryption	Digital Signatures	Proofs of knowledge, biometric traits or digital tokens + public-key enc.	ABAC and Interactive Rule Learning

The smart products analyze the manually generated rule set taking the usability and security constraints presented in Section V and produce new rule sets that are free of conflicts. In our example, the smart products present to the user *A* two automatically generated rule set options:

```
Rule Set 2
 1: If (age > 40) -> full access
 2: If (family & age > 16) -> remote and local acc
 3: If (age > 9)  -> local access
```

and,

```
Rule Set 3
 1: If (parents)  -> full access
 2: If (family & age > 16) -> remote and local acc
 3: If (age > 16) -> local access
```

It is up to *A* to decide which rule set suits her needs the best. Both rule sets look much better and concise than the manually generated rule set. However, the first rule of the Rule Set 2 is way too general (an infringement to requirement *S*1), since it gives full access rights for everyone above 40, which would include eventual guests. The last rule of Rule Set 2 is also not of her likes, since *A* would not trust a 9-year old to operate kitchen appliances (but she would trust a 12-year old). Thus, *A* picks Rule Set 3, but manually changes rules 2 and 3 to better fits her expectations. The modified rule set, Rule Set 4, is:

```
Rule Set 4
 1: If (parents) -> full access
 2: If (family & age > 12) -> remote and local acc
 3: If (age > 12) -> local access
```

A comparison between the manually generated Rule Set 1 and the interactive generated Rule Set 4 demonstrates a great improvement of the latter regarding the usability and security requirements presented in Section V. Rule Set 4 addresses the security requirements *S*1 and *S*2 since the rules are specific and meaningful. Usability requirements *U*1, *U*2, *U*3, and *U*4 are also fulfilled since there are no redundant rules, and the rules are consistent, understandable, meaningful, manageable and provide a minimum amount of rules to express the owner's security expectations.

XI. CONCLUSION

In this paper, we showed that generation of access control rule sets is the most challenging aspect for obtaining both usability and security in a smart product scenario. Other security services, such as confidentiality, integrity, and authenticity can be automated and, therefore, made fully transparent for end-users. In Table I we summarize the usability aspects and security mechanisms regarding the aforementioned security services. Based on analysis of the different AC mechanisms, the combination of a blacklist with an attribute based access control (ABAC) approach combined with an interactive rule learning (IRL) is proposed to comply with today and future needs for smart products. Hence, we first listed a series of security and usability requirements for access control rule sets. We concluded that the combination of automated rule learning with user interaction is able to meet such requirements to a secure and usable system. A design description based on FMC diagrams showed the integration of the proposed security solution in the SmartProducts framework. The design components *Access Handler* and *Security Administration* directly correlate to the aforementioned concepts of ABAC and IRL. For both components use cases were provided to demonstrate the dynamic structure of the smart products security design.

Future work is going to exploit how IRL for ABAC can be implemented to achieve the best possible results in generating usable and secure rule sets for smart products. Initially, data needs to be collected for the automated rule generation from two different sources. The first data source is composed of rules that are already pre-loaded or added by users to smart products. The second source is the actual behavior of users of smart products that can be observed by the intrusion detection component. The combined data is going to be used by the automatic rule learner to define a new set of rules that are submitted to the user for approval.

We are also going to address the processing of hierarchical data in automatic rule learning in the near future. Rule learning on hierarchical data is important to allow the users to define natural access rules. Hierarchical data provides crucial contextual information. They are commonplace in many aspects of our daily lives. For instance, business structures are mostly hierarchical, with directors, managers, and secretaries. Current automatic rule learners are not able to process hierarchical data and, therefore, they need to be extended to accept such data[2]. A final aspect is the conversion of automated generated rules to rules that are user-friendly, i.e. rules that are simple and easy to understand.

[2] There are indeed already proposals of rule learning on hierarchical data [25]. However, those are still very limited regarding the use of the hierarchical structures.

Acknowledgements

This research paper is part of research conducted by the ICT 7^{th} Framework Programme SmartProducts of the European Commission (grant n° 231204) and supported by the Center for Advanced Security Research Darmstadt (CASED).

References

[1] M. Beckerle, L. A. Martucci, and S. Ries, "Interactive access rule learning: Generating adapted access rule sets," in *ADAPTIVE 2010 : The Second International Conference on Adaptive and Self-Adaptive Systems and Applications*, ser. ComputationWorld 2010. IARIA, Nov 2010, pp. 104–110.

[2] M. Beckerle, "Towards Smart Security for Smart Products," in *AmI-Blocks'09: 3rd European Workshop on Smart Products*, 2009.

[3] L. Cranor and S. Garfinkel, *Security and Usability*. O'Reilly Media, Inc., 2005.

[4] A. Herzog and N. Shahmehri, *New Approaches for Security, Privacy and Trust in Complex Environments*, 232nd ed. Springer Boston, 2007, pp. 37–48.

[5] V. Reding, *The Future of the Internet - A conference held under the Czech Presidency of the EU*. Belgium: European Commission - Information Society and Media, 2009, ch. What policies to make it happen?, pp. 2–5.

[6] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-hashing for message authentication," 1997.

[7] F. Stajano, *Security for ubiquitous computing*. John Wiley and Sons, 2002.

[8] S. Brand, "DoD 5200.28-STD Department of Defense Trusted Computer System Evaluation Criteria (Orange Book)," *National Computer Security Center*, 1985.

[9] D. Bell and L. La Padula, "Secure computer system: Unified exposition and Multics interpretation," *MTR-2997*, 1976.

[10] D. Ferraiolo, D. Kuhn, and R. Chandramouli, *Role-based access control*. Artech House Publishers, 2003.

[11] E. Yuan and J. Tong, "Attributed based access control (ABAC) for Web services," in *IEEE International Conference on Web Services ICWS 2005. Proceedings*, 2005.

[12] I. Ray, M. Kumar, and L. Yu, *LRBAC: A location-aware role-based access control model*. Springer, 2006.

[13] J. Carbonell, R. Michalski, and T. Mitchell, *An overview of machine learning*. Tioga Publishing Company, Palo Alto, 1983.

[14] J. Fuernkranz, "Separate-and-conquer rule learning," *Artificial Intelligence Review*, vol. 13, no. 1, pp. 3–54, 1999.

[15] H. Hamed and E. Al-Shaer, "Taxonomy of conflicts in network security policies," *IEEE Communications Magazine*, vol. 44, no. 3, pp. 134–141, 2006.

[16] M. Beckerle, "Interaktives Regellernen," Master Thesis, Technische Universität Darmstadt, 2009.

[17] M. Riedmiller, "Advanced supervised learning in multi-layer perceptrons-from backpropagation to adaptive learning algorithms," *Computer Standards and Interfaces*, vol. 16, pp. 265–278, 1994.

[18] E. Yair and A. Gersho, "The Boltzmann perceptron network: A soft classifier," *Neural networks*, vol. 3, no. 2, pp. 203–221, 1990.

[19] B. Schoelkopf, C. Burges, and A. Smola, *Introduction to support vector learning*. MIT Press Cambridge, MA, USA, 1999.

[20] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 9, no. 1, pp. 3–12, 2005.

[21] S. Haykin, *Neural networks: a comprehensive foundation*, 3rd ed. Prentice Hall, 2008.

[22] D. Schreiber, Ed., *SmartProducts Public deliverable D.6.2.2: Final Architecture and Specification of Platform Core Services*, Jan 2011, retrieved: Jan 2012, from http://www.smartproducts-project.eu/media/stories/smartproducts/publications/SmartProducts_D6.2.2_Final.pdf.

[23] F. Keller and S. Wendt, "Fmc: An approach towards architecture-centric system development," in *ECBS*. IEEE Computer Society, 2003, pp. 173–182.

[24] M. Beckerle, Ed., *SmartProducts Public deliverable D4.2.2: Final Concept for Security and Privacy of Proactive Knowledge*, Feb 2011, retrieved: Jan 2012, from http://www.smartproducts-project.eu/media/stories/smartproducts/publications/SmartProducts_D4.2.2_Final.pdf.

[25] W. Cohen, "Fast effective rule induction," in *Proceedings of the Twelfth International Conference on Machine Learning*, 1995, pp. 115–123.