# XSACd - Cross-domain Resource Sharing & Access Control for Smart Environments

Konstantinos Fysarakis[1*], Othonas Sultatos[1], Charalampos Manifavas[2], Ioannis Papaefstathiou[1] and Ioannis Askoxylakis[3]

[1]Electronic & Computer Engineering Dept., Technical University of Crete, Chania, Greece
[2]Electrical Engineering and Computing Sciences Dept., Rochester Institute of Technology, Dubai, U.A.E.
[3]Institute of Computer Science, Foundation for Research and Technology – Hellas (FORTH), Heraklion, Greece

**Abstract**
Computing devices already permeate working and living environments; a trend which affects all aspects of modern everyday lives, and one which is expected to intensify in the coming years. In the residential setting, the enhanced features and services provided by said computing devices constitute what is typically referred to as a "smart home". However, the long-promised improvement in the quality of residential life cannot be realized without overcoming some significant obstacles introduced by these technological advancements. The direct interaction smart devices often have with the physical world, along with the processing, storage and communication of data pertaining to users' lives, i.e. private sensitive in nature, bring security concerns into the limelight. The resource-constraints of the platforms being integrated into a smart home environment, and their heterogeneity in hardware, network and overlaying technologies, only exacerbate the above issues. This paper presents Cross-domain Service Access Control for devices (XSACd), a framework that combines the well-studied fine-grained access control provided by the eXtensible Access Control Markup Language (XACML) with the benefits of Service Oriented Architectures through the use of the Devices Profile for Web Services (DPWS). Based on standardized technologies, it enables seamless interaction and fine-grained policy-based management of the heterogeneous embedded devices that may be found in a smart residential setting, including support for interactions between users and devices residing on different locations and networks. The proposed framework is implemented in full and its performance is evaluated on a test bed featuring relatively resource-constrained smart platforms and embedded devices.

***Keywords*:** Access Control, Web Services, Ubiquitous Computing

## 1. INTRODUCTION

In recent years, massive advancements in computing and communication technologies have led to what can only be described as a revolution in terms of how people perform the various tasks comprising their everyday lives; a revolution enabled by the ubiquitous presence of computing devices in all aspects of modern life. These major changes could not leave the residential environment unaffected, with smart homes gradually becoming a reality, i.e. homes featuring sophisticated lighting (e.g. smart light bulbs), ambient environment controls (e.g. heating, ventilation and air conditioning via smart thermostats), appliances (smart -fridge, -oven, -washing machine, -coffee makers etc.), communication systems (including smart phones), entertainment (e.g. smart TVs), and home security (smart cameras, door and window controls etc.) devices. Moreover, the residential environment borders with other ubiquitous computing applications, like smart metering and e-health, as these will have to be integrated into the smart home ecosystem. Nevertheless, as said devices typically handle personal sensitive data and often feature direct interaction with the physical world, a key factor in the wider adoption and success of these new technologies will be the effectiveness with which the various security and privacy concerns [1][2] are tackled. A necessary instrument in successfully addressing these issues is the presence of robust access control mechanisms.

This paper presents Cross-domain Service Access Control for devices (XSACd), a scheme which aims to address the above requirements by defining a policy-based Access Control (AC) mechanism based on the eXtensible Access Control Markup Language (XACML [3], an OASIS standard), thus providing the means to control access to the resources of smart home nodes, even across different networks, based on policy

constraints centrally managed by the system owner. Typical XACML deployments require the setup of complex infrastructures to enable entities' interaction and policy retrieval (e.g. via the Lightweight Directory Access Protocol, LDAP [4]); such an approach may be acceptable for corporate environments but is not suitable in the context of consumer applications and the average home user. To this end, the proposed framework leverages the benefits of Service Oriented Architectures (SOAs) by implementing key entities using the Devices Profile for Web Services (DPWS [5]), also an OASIS standard, which allows the deployment of devices aligned with the Web Services technologies, thus facilitating interoperability among services provided by resource-constrained devices. The adoption of DPWS facilitates seamless Machine-to-Machine (M2M) discovery and interactions, allowing the deployment of the framework's entities to any platform, anywhere on the home network, with minimal involvement on behalf of the user.

By combining the above technologies, new devices can easily join existing networks and offer services protected by a predefined or dynamic policy set. Based on the policy rules set by the system owner, the proposed architecture provides fine-grained AC over the plethora of devices and services that may be found in smart home environments. Thus, XSACd assists in the use of the various smart devices aiming to enhance consumers' lives, while addressing their security concerns.

A key limitation to the use of DPWS across different domains, is that its device discovery is limited to local networks (as it is based on UDP multicast messages). To address this, we also introduce proxies that, based on an external, Internet-based, broker, can enable the discovery and interaction with DPWS devices residing in other networks, in an automated manner (i.e. in a seamless way from the user's perspective).

This work is organized as follows: Section 2 provides the rationale behind this work, and Section 3 includes relevant research efforts identified in the literature. Section 4 presents the proposed framework and its key entities, along with the authors' approach to implementing the framework. Section 5 includes the performance evaluation of the developed solution on typical devices that may be found in a smart home environment. Finally, Section 6 concludes the article, containing pointers to areas that future work could explore to further enhance the presented scheme.

## 2. RATIONALE

In a typical ubiquitous-computing-enhanced residential setting, various smart devices are expected to be present on appliances (e.g. smart fridge) and automation-enabled structures (e.g. smart doors), also including environmental sensors and actuators. Moreover, these are typically complemented by purpose-built devices intended to organize, manage and enhance the functionality of the rest of the smart infrastructure, like energy monitors and control nodes (e.g. a computing system with touch-based input to allow seamless monitoring and interaction with the devices).

This heterogeneous assortment of devices will feature a variety of services, each with its own intrinsic characteristics (some being critical in terms of the residents' safety, others dealing with private sensitive data etc.), thus requiring a different protection profile. For example, all residents should be able to control the smart doors and windows of a house, but, perhaps, children should not be able to tamper with a subset of those (e.g. front door) at certain timeframes (e.g. during the night). In another scenario, visitors may have the rights to monitor the environmental sensors of the residence, but not to set the climate control at their will. Moreover, the residence owners may decide they feel alright with visitors checking the contents of their smart fridge, but they, expectedly, should not be able to add goods to the shopping lists. Assuming the presence of e-health devices in the smart home ecosystem, it is anticipated that the patient, her spouse and medical staff should be able to monitor the various readings and control the actuators that deliver the prescribed medicine, but only the latter group should have access to the service that controls the drug dosage. Moreover, it would be desirable to allow medical staff to operate on the devices remotely, to avoid unnecessary visits to the hospital. In cases where the residence is equipped with smart-metering devices, authorized power company staff should be the only ones able to adjust and/or reset the meters remotely (for billing purposes), but, nevertheless, the owners should be able to access the consumption readings as well.

Furthermore, a survey [6] on smart home users revealed that inflexibility (often forcing users to adopt solutions offered by a single manufacturer) and difficulties in achieving security constitute significant barriers to the broader adoption of pertinent technologies and devices.

From the above, and considering that, typically, the only pervasive protection mechanism present in home environments is the access to the wireless network, it is evident that strong and interoperable access control

mechanisms are required to safeguard a variety of aspects pertaining to the operation of a smart home environment. Additionally, this should be achieved in a flexible, platform-agnostic manner, acting as an enabler instead of introducing new (or further exacerbate existing) obstacles to the adoption of "smart" devices and services.

To this end, the presented framework is based on standardized mechanisms, which also allows leveraging work already carried out both in terms of Web Services as well as XACML policy definitions. DPWS can enable user-to-machine and M2M interactions in a unified manner, moving on from the current state of the field, where consumer electronics manufacturers offer a variety of proprietary protocols which are not interoperable and essentially lock-in consumers, forcing them to use a specific vendor/ecosystem. With regard to XACML, the scheme can trivially be expanded to cater for additional specific concerns, such as privacy issues and/or the handling of sensitive data (e.g. healthcare, as covered by the relevant OASIS Cross-Enterprise Security and Privacy Authorization profile for XACML v2.0 [7]).

## 3. BACKGROUND & RELATED WORK

### *3.1 Service-Oriented Architectures (SOAs)*

SOAs evolved from the need to have interoperable, cross-platform, cross-domain and network-agnostic access to devices and their services. This approach has already been successful in business environments, as web services allow stakeholders to focus on the services themselves, rather than the underlying hardware and network technologies.

The Devices Profile for Web Services (DPWS) specification defines a minimal set of implementation constraints to enable secure Web Service messaging, including discovery, description, interactions and event-driven changes on resource-constrained devices. DPWS was introduced in 2004 and is now an OASIS open standard (at version 1.1 since July 2009). It employs similar messaging mechanisms as the Web Services Architecture (WSA), with restrictions on complexity and message size, allowing the provision of totally platform- and language-neutral services, similar to those offered by traditional web services, allowing system owners to leverage the SOA benefits across heterogeneous systems that may be found in the various smart environments (residential, enterprise etc.).

It should be noted that DPWS was originally conceived and introduced as a successor to Universal Plug and Play (UPnP [8]), but the lack of backward compatibility meant that such a transition has not taken place yet. While UPnP is the basis of many solutions proposed in the literature for the integration and management of smart home devices, DPWS features significant improvements, justifying the migration to this newer specification. In more detail, as DPWS is built around the OASIS Web Services (WS-) stack, it offers significant benefits in terms of operational and development aspects, as well as in device-to-device and user-to-device communication. Deploying Web Services on devices allows the use of a single stack for local network and Internet interactions with devices and other Web Services. Moreover, it helps take advantage of pre-existing tools and development know-how and, most importantly, it allows leveraging the extensive work already carried out in defining the various WS-* family of protocols (e.g. for security, an important aspect for which UPnP is lacking provisions). Thus, DPWS-enabled devices feature superior scaling and seamless integration capabilities, both in the context of smart home (e.g. smart appliances, energy monitoring or smart metering) and enterprise (e.g. plant floors or enterprise-wide information systems) environments and the Internet (e.g. regular W3C-specified Web Services). Scalability is a key area where UPnP is found wanting, as it is only suitable for small local networks. Comparatively, the main disadvantage of DPWS is the existence of numerous specifications (protocols, bindings etc.), which have not been consolidated yet.

The SOA-based approach of DPWS acts as an enabler, with added potential stemming from the enhanced real-time monitoring and control features usable over the whole smart infrastructure. By facilitating the migration of low level data (e.g. sensing data) into higher level contexts (e.g. user privacy or energy usage preferences, also including knowledge extraction from aggregated data), new types of services are made possible. These new types of enhanced features and services are expected to be vital for future end-users as well as smart home deployments.

In this context, the work of Leong et al [9] presents a rule-based framework for heterogeneous smart-home systems management. Their work focuses on the use of SOAP for interoperability and uses an Event-Condition-Action (ECA) mechanism for machine-2-machine interactions and orchestration of the devices. The SOAP-based interoperability framework has been further extended by Perumal et al [10] with the addition of

a service stub to facilitate the addition of new devices and a database module to handle the queries of the SOAP messages (including home service functions, operation logic and access to other local or remote databases).

SOA-DOS [11] is a SOA-based distributed operating system proposed in the relevant literature, aiming to manage all embedded devices in a home network and facilitating interoperability between the various systems. The work manages to provide a SOA-based solution that is also applicable to very resource-constrained platforms (like sensor nodes), but deviates from standardized mechanisms, e.g. resorting to the use of the JSON [12] format instead of XML for data exchange.

The use of SOA concepts to tackle the dynamic and heterogeneous nature of home-control applications has also been proposed by Bourcier et al [13]. The authors introduce an implementation of their approach based on open source, standardized platforms, providing bridges to seamlessly integrate disparate devices (including DPWS devices) and their services into their home control infrastructure.

The DPWS stack also forms the basis of iVision [14], a purpose-built hardware platform used to add context-awareness to a service architecture for controlling home appliances, and its accompanying architecture. In the above work, the context information extracted by the iVision camera and all the necessary smart home appliance communications are exposed as web services using DPWS.

The use and benefits of DPWS have also been studied extensively in the context of various other applications areas, including automotive and railway systems (Venkatesh et al [15]), industrial automation (Cucinotta et al [16]), eHealth (Pöhlsen et al [17]), and wireless sensor networks (Dohndorf et al [18]). All of the above are positive indicators for the future of the technology chosen as the underlying implementation and communication mechanism for the presented framework, and its potential for ubiquitous adoption.

### *3.2 Policy-based Access Control*

Among the studied access control schemes proposed for systems with different requirements and properties, a cross-platform solution that meets the requirements of all types of embedded systems and provides interoperability (which is crucial for next-generation pervasive computing devices), is based on eXtensible Access control Markup Language (XACML) policies. XACML is an XML-based general-purpose access control policy language used for representing authorization and entitlement policies for managing access to resources. However, it is also an access control decision request/response language. As such, it can be used to convey policy requirements in an interoperable and secure manner, if appropriately deployed. XACML is used to convey policy requirements in a unified and unambiguous manner, which fits well into the model of a network of heterogeneous embedded systems where access to resources is provided by nodes as a service. Thus, XACML defines the structure and content of access requests and responses exchanged among access control entities. In this work, the typical policy based access control architecture, combined with XACML, is mapped to a SOA network of nodes to provide protected access to their distributed resources.

A survey of the literature reveals a wealth of related work, including various diverse approaches and attesting to the applicability of XACML in the context of smart homes.

Kim et al [19] have proposed the use of an OSGi (Open Services Gateway initiative) -based framework to integrate heterogeneous smart-home devices and services. The proposed framework also includes an access control model, combining the XACML mechanisms with OSGi services to appropriately create the queries that will be forwarded to the entity responsible for access control decisions (i.e. the Policy Decision Point, PDP). While the proposed approach theoretically supports a variety of protocols (including DPWS devices), the presented analysis and proof of concept implementation are mainly based on UPnP, a protocol lacking in many respects, already noted above (e.g. security & scalability). Furthermore, the performance of the proposed mechanisms – an important aspect considering the resource-constraints of many smart devices – is not evaluated.

Another approach found in the literature is the use of a "Reference Monitor" entity [20], i.e. a home gateway, whose goal is to provide and enforce a collection of access control policies, aiming to help satisfy a user's access, convenience, and privacy requirements. The access control policies are defined using XML, but deviate from the standard XACML syntax and architecture.

Busnel et al [21] present a case study for remote healthcare assistance in smart homes. Most of the smart home security & dependability requirements are discussed extensively, identifying the use of SOAs along with XACML as the most applicable technologies to fulfill these requirements. An XACML-based authori-

zation solution is applied using the security pattern approach to satisfy security requirements typically existing in such environments. This work presents the outline of such a framework, but not an actual implementation of the SOA and XACML mechanisms, nor a performance evaluation. The resource-constrained nature of the target devices and the use of appropriate security mechanisms do not appear to have been considered during the design phase.

Researchers have also studied the use of access control mechanisms to safeguard the users' privacy, a key concern in the context of smart environments. Faravelon et al [22] outline such an architecture in the context of SOA-enabled pervasive environments, using a medical scenario as a test case. The interoperability with DPWS is considered, among other SOA technologies, but a non-standardized approach is adopted for the access control functionality.

Privacy issues have also been considered by Jung et al [23], who have presented a generic concept of access control for home automation gateways, aiming to safeguard the privacy and security of users and their data. The scheme is based on a customized SOAP message structure that integrates XACML attributes within SAML-based access token. However, the initial, theoretical evaluation of the proposed scheme indicates that this approach is quite costly (especially in terms of packet size), which questions its applicability in the context of embedded smart home devices. The authors acknowledge this drawback and indicate it will be investigated, as future work, on actual platforms.

Müller et al [24] have also proposed the combined use of DPWS with XACML, but focusing on end-user content (e.g. the distribution of multimedia files). They also use proxies to establish trust relationships across smart home domains but the authors did not exploit DPWS in the implementation and deployment of the XACML architecture.

## 4. Proposed Model & Implementation Approach

This section aims to detail the basic components of the proposed solution, as well as the toolkits chosen to develop the proof of concept implementation.

### *4.1 Main entities*

In the proposed framework, the following key entities are present and can be deployed on different smart home nodes, depending on their role and resources:

- *Policy Enforcement Point (PEP):* Makes decision requests and enforces authorization decisions. This is expected to be present in every smart device (appliances, sensors, e-health devices, energy monitoring or smart metering devices etc.) which provides its resources to the end users, and which need to be protected by the active policy set.
- *Policy Decision Point (PDP):* Evaluates requests against applicable policies and renders an authorization decision. It is expected to be deployed on more feature-rich nodes, typically a personal computer or an embedded system that acts as a controlling node for the whole smart home infrastructure
- *Policy Administration Point (PAP) & Policy Information Point (PIP):* The former creates and manages policies or policy sets, while the latter acts as a source of attribute values. These two entities will typically be deployed on the same feature-rich node, facilitating direct interaction with end-users (e.g. home owners). A desktop computer or a laptop are good candidates for this role.
- *Cross-domain Proxy:* This entity is responsible for catching all discovery messages of the network, processing their contents and transmitting them to other networks, also transmitting to the local network all messages received from other domains.
- *Broker:* This is the main entity through which all cross-domain traffic is routed. The broker is responsible for distributing messages to all interested clients (i.e. the proxies) based on a message's topic. To this end, all proxies have to subscribe to the Broker.

As is evident from the above, and considering that nodes embedded in a smart home may not have the computing resources to accommodate expensive mechanisms, the core decision process is undertaken by more powerful nodes expected to operate within the node's trusted environment. Such an approach allows requests to be directly addressed to the node in question, while maintaining the capability to centrally manage and control access to these nodes. An overview of the architecture can be seen in Figure 1.

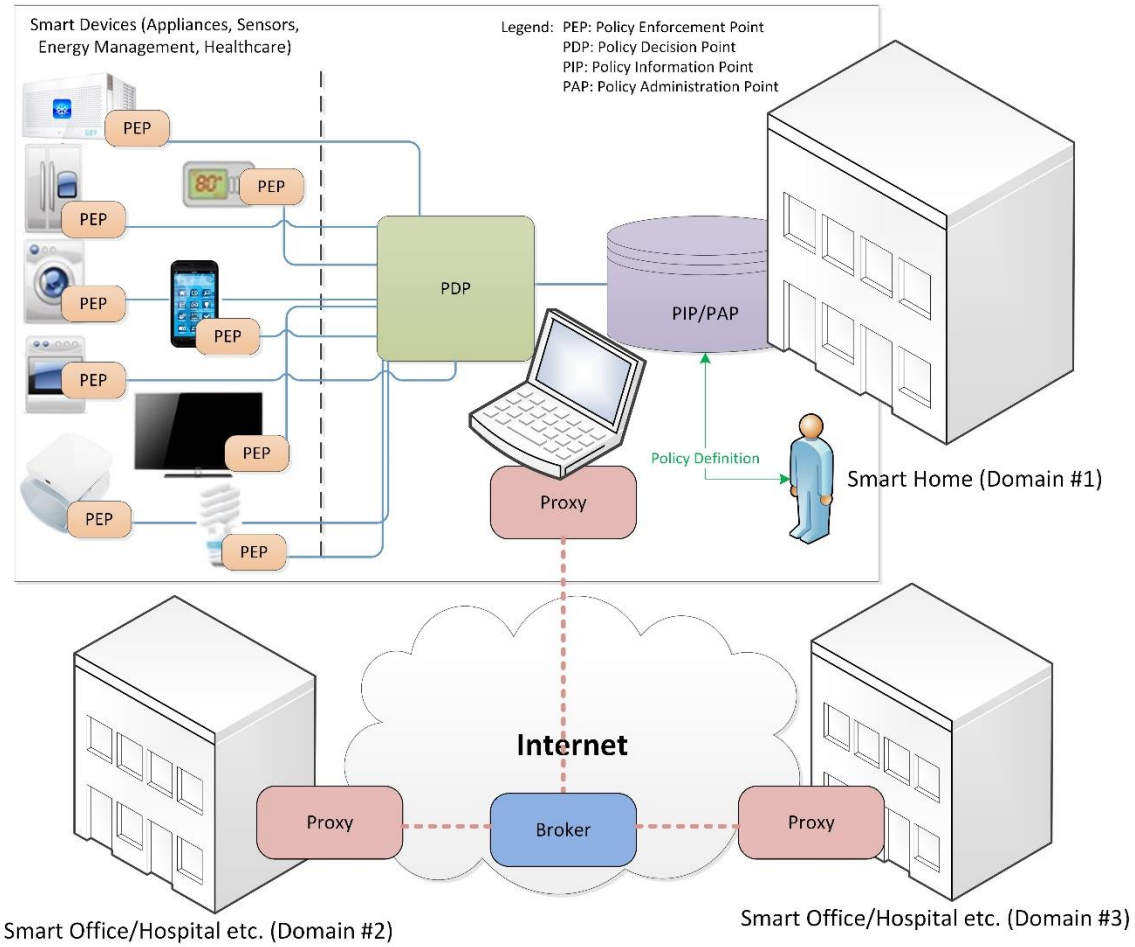


**Figure 1. Smart Home Access Control architecture & cross-domain communication. Main entities.**

There are various open-source and commercial implementations of XACML that could be used as a basis for the AC entities needed to implement the proposed framework. The authors chose to use Sun's Java-based XACML implementation, as it remains popular among developers and is actually the basis of various current open-source and commercial offerings.

### *4.2 DPWS implementation of the XACML entities*

All of the framework's entities are exposed to the network using DPWS, thus leveraging the benefits provided by SOAs. There are a variety of APIs available for DPWS development, including the tools provided by the Web Services for Devices (WS4D) initiative and the SOAs for Devices (SOA4D) toolkits, based on various programming languages (C, C++, Java etc.) and each featuring its own intrinsic characteristics. Nevertheless, when focusing on key features such as code portability, deployment on heterogeneous platforms, support for IPv6 (necessary for ubiquitous computing applications) and, most importantly, active development and support of the tools, WS4D-JMEDS currently stands out as the most attractive choice. It is the most advanced and active work of the WS4D initiative, supporting most of the existing DPWS features and providing portability to a wide range of platforms; it is also the authors' toolkit of choice to develop the DPWS entities presented in this work.

Using the above API and exploiting the features of DPWS, the XACML functionality can be exposed to the home network (and the Internet, if needed), allowing the seamless discovery and communication of the framework's entities, regardless of the device where they may be deployed. In more detail, the XACML features are exposed as follows:

#### *4.2.1 PEP to PDP implementation*

The Policy Enforcement Point must reside on every device with resources that must be protected from

unauthorized access. Other than the functional elements of the devices which the framework intends to protect (e.g. access to its sensors), two extra operations must be present on each DPWS device. These operations, in essence, constitute the PEP functionality and its communication with the PDP. The latter acts as a DPWS client which accesses these PBAC-specific operations.

More specifically, the first operation is the "SAREvent" (Service Access Request Event), referring to an operation following the WS-Eventing specification to which devices can subscribe. When fired, the operation outputs "SAROut", a message which includes all the information the PDP needs to have in order to evaluate a request (i.e. Subject, Action and Resource). The second operation is "PDPResponse" (Policy Decision Point Response), which is invoked by the PDP to relay an answer to a pending access request.

*4.2.2 PDP to PIP/PAP implementation*

In terms of the discovery and information exchange that must take place between infrastructure entities (PDP, PIP, PAP), an extra operation must reside with the entity that stores the active policy set (namely the PIP/PAP). This extra operation is named "PIPOperation" (Policy Information Point Operation). It features an input for the request issued by the PDP (requesting all applicable policy rules), and an output containing all the pertinent information (i.e. policies and rules) that the PIP has identified.

*4.3 Event sequence*

The above DPWS operations and the sequence of events that take place when an access request is received for a protected resource are depicted in Figure 2.

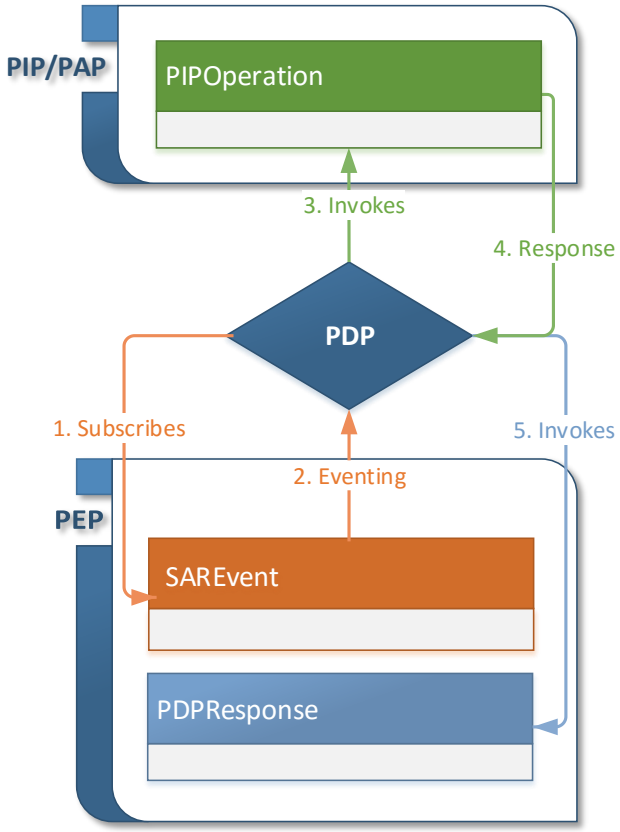


**Figure 2. DPWS implementation of the XACML mechanisms.**

In more detail, the PDP is implemented as a DPWS client, constantly monitoring the network for "Hello" messages transmitted by DPWS devices as they initialize. Whenever it discovers a PEP-equipped device, it automatically subscribes to its "SAREvent" operation.

Each time a user tries to access a resource on one of these AC-protected devices, the "SAREvent" is fired, notifying the PDP that a request has to be evaluated, and transmitting all the information required for XACML to make such a decision. This information includes the *Request ID* (a counter of policy decision requests issued by the specific device), the Client's identifier, e.g. IP and/or username (*Subject*), the Invoked

Operation (*Action*) and the Device's UUID, i.e. its Universally Unique Identifier (*Resource*).

The above data is used by the PDP to generate an XACML request, i.e. a request in a form that can be evaluated by the XACML decision engine. This process also involves the PDP invoking the "PIPOperation" of the PIP/PAP, in order to retrieve the applicable policies.

Based on the information included in the PEP request and the pertinent policies retrieved from the PIP/PAP, the PDP can then decide if the user's pending request is authorized or not. This decision is conveyed to the PEP by invoking its "PDPResponse" operation.

Finally, based on the decision received, the PEP either grants or denies access to the device's resource that the user initially tried to access.

### *4.4 DPWS Proxy*

On its own, DPWS does not support cross-domain communications, since device discovery is based on a UDP multicast protocol, WS-Discovery. So, without the use of a purpose-built proxy, all the above entities must be on the same network in order to operate since DPWS device discovery is based on a UDP multicast WS-Discovery [25]. Our proxy implementation is not based on a central server that catches all "hello" and "bye" messages as is typical for such proxies; this presumes that at every discovery proxy has a list of known devices and that proxies communicate each other to "share" devices, which is not really usable.

XSACd's proxy uses MQTT [26], a publish-subscribe OASIS standard protocol, as the backbone of communication between networks. An MQTT Broker, deployed somewhere on the Internet, is responsible for handling and organizing all communications between the various domains and their corresponding proxies. In every local network a MQTT client is deployed which has two roles:

- catch all discovery messages of the network, handle the info and transmit them at the other DPWS networks and
- transmit to the current network all messages that are published from other networks.

Catching all messages from the network is done by adding a membership to a predefined address, namely 239.255.255.250:3702, which is the UDP broadcast and default port of DPWS discovery default settings. Then every SOAP xml message is parsed; if it is a "hello" message the IP of the message is changed from the 192.168.*.* ,which is normally used in a local network, to the external IP, also using NAT-PMP (if the network's router supports it) the local port is forwarded, so that external communication is achieved. Then the changed "hello" message is transmitted to all the other DPWS proxy subscribers. When a client from a local network searches (sends a probe) for a device, the proxy saves the ID of the message and also the information of the sender. That way, if a device is found that meets the search criteria (i.e. a probe match is received), it will only transmit the response to the local network that made the search. Moreover, the sender information is kept because, when a client sends a probe message, it waits for the probe response at the same port. Moreover, as every message ID is saved, duplicate transmissions are eliminated.

It must also be mentioned that for multi-home communication to be fully supported, after the device discovery (facilitated by the DPWS proxy), the client asks the target device of its metadata, provided in the form of a WSDL [27] file. In order to have a successful communication, the WSDL file must contain the external IP of the device. This was not supported in any available DPWS implementation, so we modified our custom DPWS implementation in the Node.js programming environment [28], allowing us to track the GetMetadata message: if it originates from the local network, it uses the local IP (192.168.*.*) at the WSDL, otherwise it uses the router's external IP. Any standard MQTT Broker could be used to interact with the XSACd proxies. For the proof-of-concept implementation, we opted for the Mosquitto open source message broker [29].

Figure 3 depicts a simplified view of the above, along with the steps that take place during normal operation; in this example, when a device from the local network issues a probe match and gets a reply from a device deployed in another domain. All external traffic is routed through the local router in a typical home deployment; to this end, the proof-of-concept implementation automates port forwarding.

Compared to the proxy presented by Müller et al [24], the XSACd approach has various benefits. Scaling is an issue with the previous work, as in the case of N networks, each proxy must communicate with N-1 other proxies. So, for example, a proxy must send its probe match request to N-1 other networks, initiating an equal number of connections. In our approach, each proxy only needs to communicate with the MQTT Broker, which is responsible for disseminating all DPWS messages to the other registered proxies. Moreover, there is added complexity in that the proxy has to tamper with the metadata files, whereas in XSACd all

devices properly formulate their metadata files by themselves. Finally, the authors propose different types of proxies (client proxy & server proxy), depending on the network's role, whereas an XSACd proxy can accomplish both roles simultaneously.

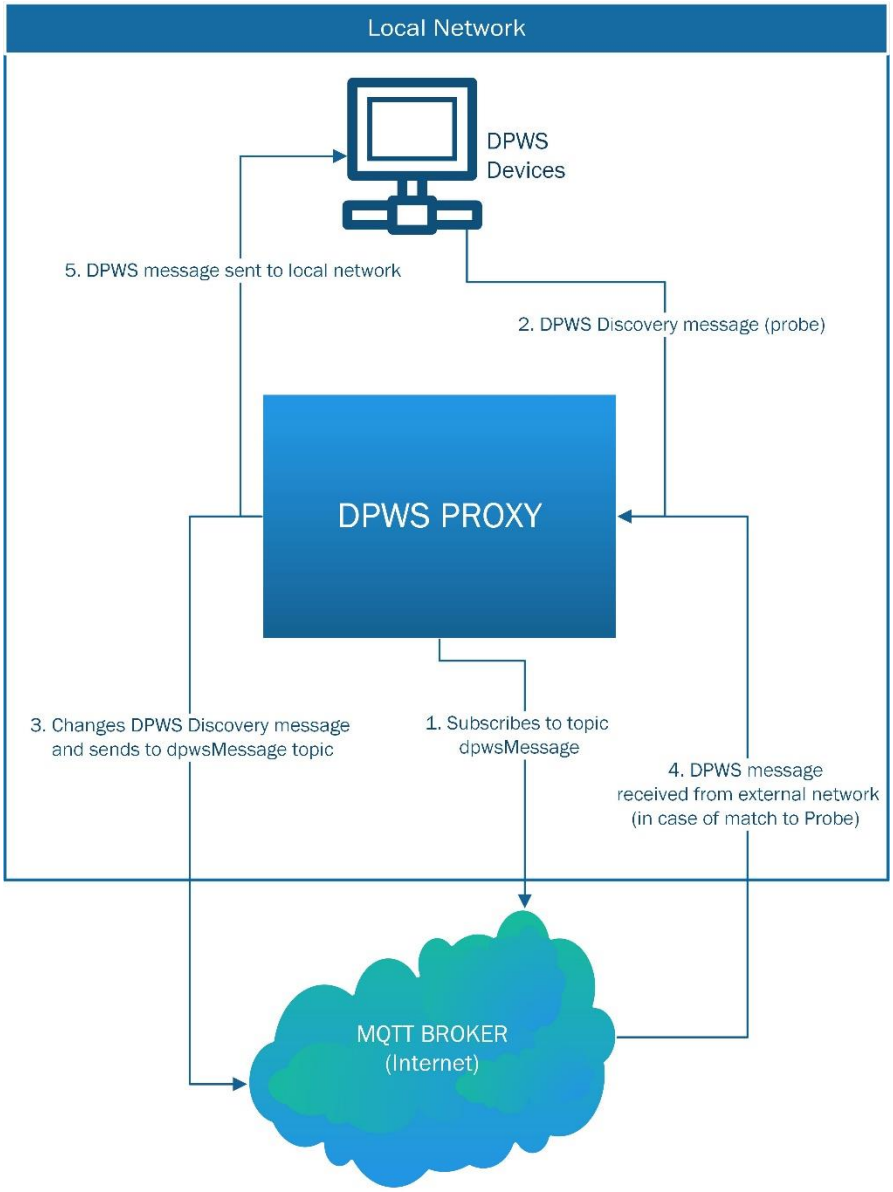


**Figure 3. XSACd cross-domain proxy implementation and main steps. Simplified view.**

*4.5 Security considerations*

The effectiveness of any access control mechanism can easily be compromised unless appropriate security mechanisms are deployed to protect policy messaging. A malicious entity would otherwise be able to eavesdrop, replay or tamper with the access control messaging, overriding the offered protection to provide access to unauthorized entities or denying access to authorized ones. When feasible, deployments over trusted and/or secure networks (e.g. over a Virtual Private Network, VPN) can address most of these concerns, but an alternative mechanism has to be considered for deployments where these provisions are not realistic.

The Web Services Security Specification (WS-Security or WSS, [30]) is part of the WS-* family of specifications published by OASIS. The protocol specifies integrating security features in the header of SOAP messages. Working in the application layer, it ensures the end-to-end integrity and confidentiality of SOAP messages. Therefore, an alternative of the proof-of-concept implementation was developed as well, adopting

the security mechanisms specified in WSS. These mechanisms authenticate entities and safeguard the integrity and confidentiality of the policy messaging exchanged by the framework's entities.

In terms of protecting the communication of the proxies with the MQTT broker, there are various options that could be explored. As of version 3.1, MQTT supports the use of a username and password to secure the communication with the broker. There is also the option of using web sockets and secure web sockets .Moreover, since MQTT is based on TCP, one could always opt for the use of more typical socket security mechanisms, such as SSL/TLS, but this would add a notable amount of processing overhead. In this proof-of-concept we used a symmetric mechanism based on AES/CCM [31] to encrypt every payload sent from the publisher to the subscribers. Thus, a username and password is used to secure access to the MQTT broker and, moreover, the messages exchanged are encrypted, to protect them from eavesdropping and tampering (using the authenticated encryption mechanisms of AES/CCM).

## 5. Performance Evaluation

The use of platform-agnostic technologies (i.e. DPWS and Java) enables the proposed framework to be deployed, by design, on a variety of platforms and operating systems. However, in order to realistically assess the performance of the proposed framework and its impact on the target devices, the developed entities have to be deployed on devices expected to be present in smart home environments.

Therefore, the infrastructure entities, namely the PDP and PIP/PAP, were deployed on a laptop (quad core CPU at 2.6GHz, 4GB RAM), as a personal computer is typically available in home environments and is expected to act as a management hub through which the residents monitor and control their smart residence. A total of 50 policies were stored in the policy repository, which the authors considered a realistic approximation of the number of policies needed, considering the relatively limited number of devices expected to reside in a smart home. Tests were also carried out with 500 policies, to assess the impact more policies would have on the framework's performance.

Regarding the target platforms – i.e. the platforms featuring the services that need to be protected – the authors chose to use relatively resource-constrained smart embedded devices (600MHz low power single core CPU, 512MB RAM) running the Android open source operating system for mobile devices. Such operating systems are already found in many smart commercial appliances (e.g. smart fridges) offered by the various consumer vendors. Moreover, their adoption is expected to spread as more sophisticated home devices become available to end users; thus, the above platform can be considered a realistic choice for evaluating the performance of the proposed mechanisms.

The DPWS device deployed on the smart platform not only featured the access control related operations (as depicted in Figure 2) but also featured three simple operations, emulating part of the functionality of a smart appliance (e.g. smart fridge). Via the above operations, the user can get the current temperature, subscribe to a service that periodically informs of said temperature and also set the desired temperature when needed. A basic touch GUI was developed for this device, which can be seen in Figure 4.

A client application was also developed for testing purposes; the "Smart Home Browser". This application is deployable on various end devices (personal computers, smart phones or tablets) and allows users to discover and control the various DPWS-enabled smart appliances (to get the current contents of the smart fridge, to subscribe to the power consumption readings provided by the smart metering device etc.).

A screenshot of the Smart Home Browser prototype implementation appears in Figure 5.

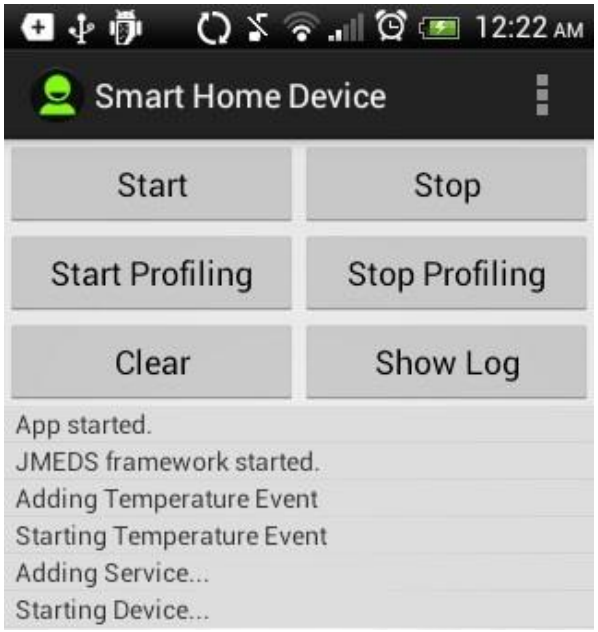


**Figure 4. Screen capture of the (PEP-protected) DPWS test device deployed on the touch-enabled smart platform.**

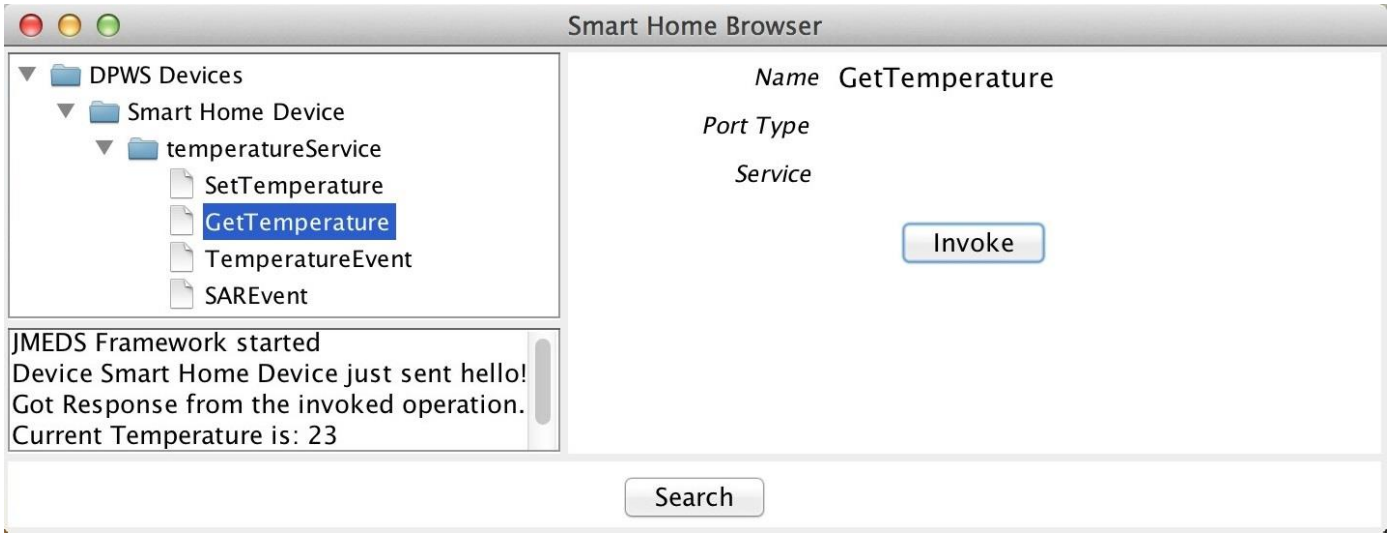


**Figure 5. The "Smart Home Browser"; an application developed to facilitate the discovery of DPWS devices and provide access to their hosted services.**

A command line-only variation of this client, programmed to automatically invoke operations and record response times, was developed for benchmarking purposes. This benchmark client was used to evaluate the performance of three setups: a simple DPWS device with no PEP implemented (i.e. with direct access to its services), a DPWS device protected by the presented access control entities communicating in plaintext, and a third setup with the entities' communications being protected via WS-Security. This allowed the authors to separately assess the impact of the access control functionality and the impact of the security mechanisms that may be needed to protect the policy messaging in some deployments.

In addition to the client-side measurements, the CPU and memory utilization was also monitored on both the personal computer that hosted the PDP and PIP/PAP as well as on the PEP-equipped smart device. Furthermore, two different usage scenarios were investigated: In the first scenario, the client issued 100 concurrent requests to invoke the services, allowing the investigation of the performance under heavy load conditions. The results appear in Figure 6.

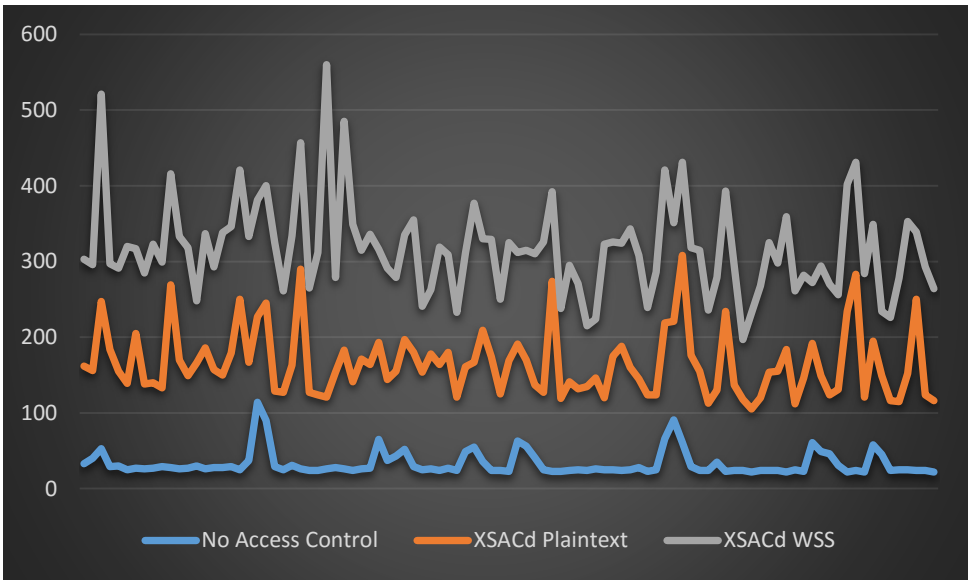


**Figure 6. Client-side response time for 100 concurrent requests (in ms).**

Investigating a second use case scenario, we set the benchmark client to issue 20 requests, one every 3 minutes, emulating more realistic usage conditions in the context of a smart home environment. The results of the above assessments in terms of the average response time (i.e. the time the user has to wait before she receives the data she intended to access) are depicted in Figure 7. In both usage scenarios, the overhead of the access control mechanism are considered acceptable. The impact of the WSS protection is significant in cases of infrequent requests (as in the second scenario, where the connections close between the request timeouts and, thus, have to be reinitiated).

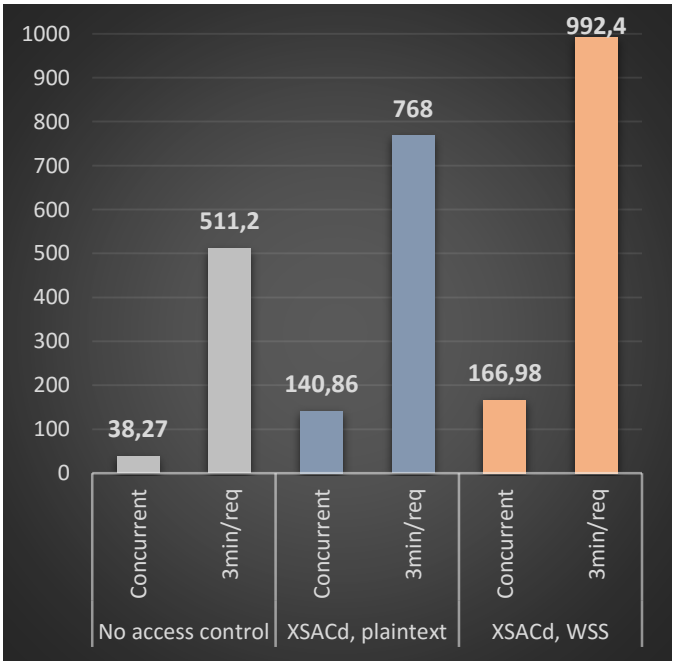


**Figure 7. Average client-side response time (in ms) for the investigated deployments and usage scenario.**

In terms of the resources consumed on the target, PEP-protected device, and focusing on the most demanding scenario (i.e. concurrent requests), profiling indicated a mild footprint during tests, even in the case of

the relatively resource-constrained smart platform used in this setup. Average memory consumption is presented in Figure 8, where the overhead of the access control mechanisms appears trivial compared to the simpler DPWS device.

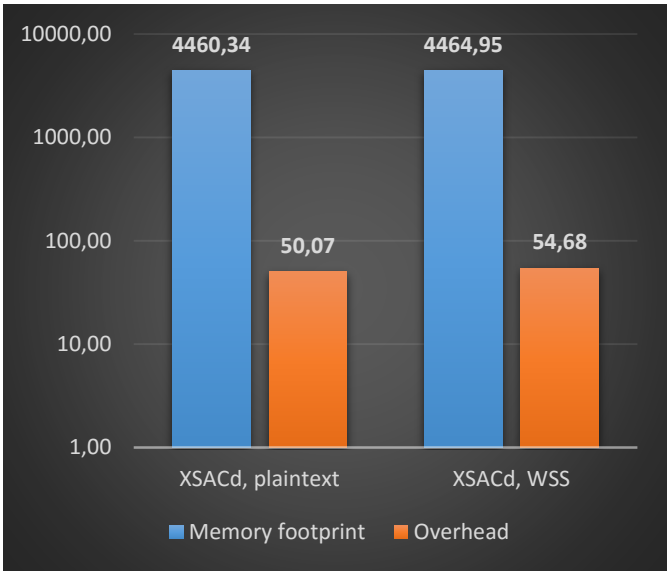


**Figure 8. Memory footprint (in kB, logarithmic scale) on the PEP-protected device, including the overhead compared to the simple DPWS device with no access control protection.**

The average CPU load was inversely proportional to the client response times (depicted in Figure 7); when the device has to wait for a reply from the framework (i.e. the PDP) before serving the client, its CPU load is expectedly lower. The recorded values were 11.6%, 9.3% and 8.4% for no access control, XSACd and XSACd with WSS respectively. The same ranking was also documented when monitoring average transmission (TX) and reception (RX) rates on the target device – see Figure 9. The most taxing scenario network-wise was that of the device with no access control, but in all cases the data rates were relatively low, with the lowest recorded value being 16.13kB (average TX of XSACd, WSS device) and the highest being 26.5kB/sec (average RX of DPWS device without access control).

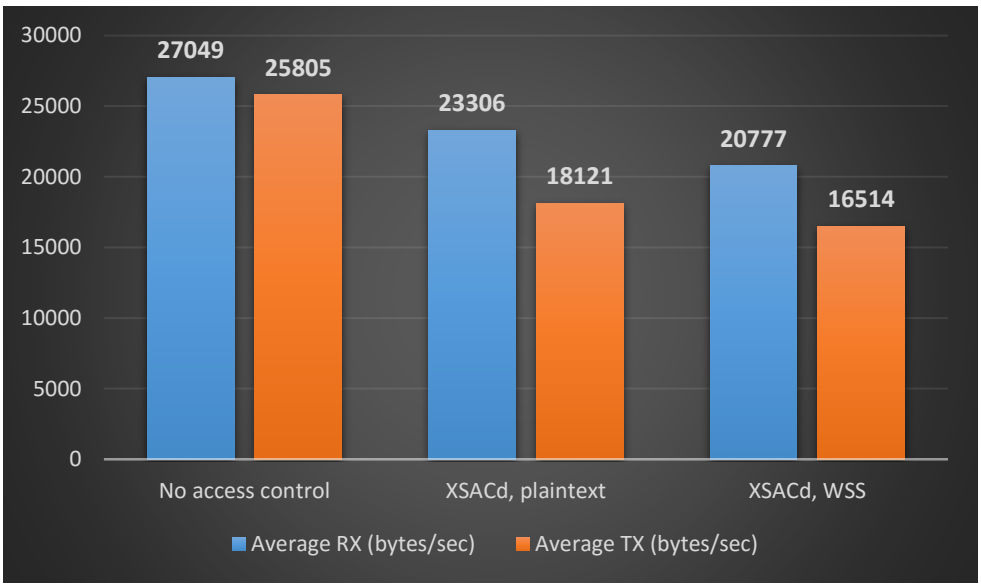


**Figure 9. Network throughput on target device during tests.**

The number of stored policies may significantly affect the performance of the access control system, to the point where the response time overhead becomes prohibitive [24]. This was taken under consideration during development and, thus, XSACd's PIP stores policies in the form of a hash table residing in memory. The effectiveness of this approach was validated during benchmarks: increasing the number of stored policies from 50 to 500 increased the response time by 7,37% in the first scenario (i.e. consecutive requests) and by just 1,7% in the more realistic second scenario (i.e. a request every 3 minutes).

Finally, to assess the load that the DPWS proxy might impose on the device it will be deployed on, the proof-of-concept implementations of the proxies were deployed on two Beagleboard-xM embedded devices (1GHz ARM Cortex-A8 processor, throttled at 600MHz during testing, 512MB DDR2 RAM, a minimal Linux-based operating system), each residing on a different geographical area and communicating via the Internet. A screenshot of a command line remote connection to the proxy appears in Figure 10.


**Figure 10. XSACd cross-domain proxy. Command line remote connection to the Beagleboard-xM embedded platform.**

Even though these test-bed platforms were relatively resource-constrained, their CPU load during testing was minimal (below 10% when handling messages, otherwise idle). The memory footprint of the application is relatively stable at around 16MB, of which around 6,6MB are occupied by the various libraries used in its implementation.

The MQTT Broker used was just a standard implementation of Mosquitto [29], thus assessing its performance is beyond the scope of this work.

## 6. Conclusions & Future Work

This paper presented XSACd, a framework that leverages the benefits of SOAs, and DPWS specifically, to enable the integration of well-studied fine-grained and adaptable access control provided by XACML into smart homes and other smart environments, often residing across different networks. The intrinsic requirements of the smart home environment, its users and the often resource-constrained nature of its devices fundamentally affected the choice and implementation of the standardized mechanism that form the basis of this work. Thus, XSACd's entities are platform-agnostic, lightweight and interact seamlessly, minimizing the home users' involvement in deploying, setting up and maintaining the system. The proxies responsible for the communication across different domains, require no interaction on behalf of the users, offering automated discovery and interactions with the target devices across the Internet.

Nevertheless, home owners will be responsible for defining some parameters of the active policy set, depending on their requirements and preferences. Thus, an important aspect to be investigated is the provision of user-friendly interfaces for specifying access control policies, e.g. using a GUI with easy to use drop-down menus and tick boxes or having the user answer simple questions, automatically translating the user input to

policies.

Lastly, while this paper focused on authorization aspects of ubiquitous smart devices, another important building block is the user authentication. To this end, the integration of XSACd with the work we presented in [32], will be investigated in the future.

## ACKNOWLEDGMENTS

Part of this work was funded by the General Secretarial Research and Technology (G.S.R.T.), Hellas under the Artemis JU research program nSHIELD (new embedded Systems arcHItecturE for multi-Layer Dependable solutions) project. Call: ARTEMIS-2010-1, Grand Agreement No: 269317.

## REFERENCES

[1] D. Sauveron, K. Markantonakis, C. Verikoukis, Security and resilience for smart devices and applications, EURASIP J. Wirel. Commun. Netw. 2014 (2014) 123. doi:10.1186/1687-1499-2014-123.

[2] C. Manifavas, K. Fysarakis, A. Papanikolaou, I. Papaefstathiou, Embedded Systems Security: A Survey of EU Research Efforts, Secur. Commun. Networks. 8 (2015) 2016–2036. doi:10.1002/sec.1151.

[3] B. Parducci, H. Lockhart, E. Rissanen, eXtensible Access Control Markup Language (XACML) Version 3.0, OASIS Stand. (2013) 1–154. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf (accessed September 1, 2015).

[4] J. Sermersheim, Lightweight Directory Access Protocol (LDAP): The Protocol, Internet Eng. Task Force. (2006) 1–69. http://www.ietf.org/rfc/rfc4511.txt.

[5] D. Driscoll, A. Mensch, T. Nixon, A. Regnier, Devices profile for web services version 1.1, OASIS Stand. July. 1 (2009)

[6] A. Brush, B. Lee, R. Mahajan, Home automation in the wild: challenges and opportunities, in: Proc. SIGCHI Conf. Hum. Factors Comput. Syst., 2011: pp. 2115–2124. doi:10.1145/1978942.1979249.

[7] D. DeCouteau, M. Davis, D. Staggs, OASIS Cross-Enterprise Security and Privacy Authorization (XSPA) Profile of XACML v2.0 for Healthcare Version 1.0, OASIS Stand. Specif. (2009) 1–21. http://docs.oasis-open.org/xacml/xspa/v1.0/xacml-xspa-1.0.pdf (accessed September 1, 2015).

[8] A. Donoho et al., UPnP Device Architecture 2.0, Architecture. (2015) 1–196. http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v2.0.pdf (accessed September 1, 2015).

[9] C.Y. Leong, A.R. Ramli, T. Perumal, A rule-based framework for heterogeneous subsystems management in smart home environment, IEEE Trans. Consum. Electron. 55 (2009) 1208–1213. doi:10.1109/TCE.2009.5277977.

[10] T. Perumal, A. Ramli, C. Leong, Interoperability framework for smart home systems, IEEE Trans. Consum. Electron. 57 (2011) 1607–1611. doi:10.1109/TCE.2011.6131132.

[11] A. Sleman, R. Moeller, SOA distributed operating system for managing embedded devices in home and building automation, IEEE Trans. Consum. Electron. 57 (2011) 945–952. doi:10.1109/TCE.2011.5955244.

[12] T. Bray, RFC7159: The JavaScript Object Notation (JSON) Data Interchange Format, Internet Eng. Task Force Req. Comments. (2014). http://tools.ietf.org/html/rfc7159 (accessed September 1, 2015).

[13] J. Bourcier, C. Escoffier, P. Lalanda, Implementing Home-Control Applications on Service Platform, in: 2007 4th IEEE Consum. Commun. Netw. Conf., IEEE, 2007: pp. 925–929. doi:10.1109/CCNC.2007.187.

[14] R.R. Igorevich, P. Park, J. Choi, D. Min, iVision based Context-Aware Smart Home system, in: 1st IEEE Glob. Conf. Consum. Electron. 2012, IEEE, 2012: pp. 542–546. doi:10.1109/GCCE.2012.6379904.

[15] V. Venkatesh, V. Vaithayana, P. Raj, K. Gopalan, R. Amirtharaj, A Smart Train Using the DPWS-based Sensor Integration, Res. J. Inf. Technol. 5 (2013) 352–362. doi:10.3923/rjit.2013.352.362.

[16] T. Cucinotta, A. Mancina, G.F. Anastasi, G. Lipari, L. Mangeruca, R. Checcozzo, et al., A Real-Time Service-Oriented Architecture for Industrial Automation, IEEE Trans. Ind. Informatics. 5 (2009) 267–277. doi:10.1109/TII.2009.2027013.

[17] S. Pöhlsen, S. Schlichting, M. Strähle, F. Franz, C. Werner, A DPWS-Based Architecture for Medical Device Interoperability, in: O. Dössel, W. Schlegel (Eds.), World Congr. Med. Phys. Biomed. Eng. Sept. 7 - 12, 2009, Munich, Ger. SE - 22, Springer Berlin Heidelberg, 2009: pp. 82–85. doi:10.1007/978-3-642-03904-1_22.

[18] O. Dohndorf, J. Kruger, H. Krumm, C. Fiehe, A. Litvina, I. Luck, et al., Towards the Web of Things: Using DPWS to bridge isolated OSGi platforms, in: 2010 8th IEEE Int. Conf. Pervasive Comput. Commun. Work. (PERCOM Work., IEEE, 2010: pp. 720–725. doi:10.1109/PERCOMW.2010.5470527.

[19] J.E. Kim, G. Boulos, J. Yackovich, T. Barth, C. Beckel, D. Mosse, Seamless Integration of Heterogeneous Devices and Access Control in Smart Homes, in: 2012 Eighth Int. Conf. Intell. Environ., IEEE, 2012: pp. 206–213. doi:10.1109/IE.2012.57.

[20] E.-A. Cho, C.-J. Moon, D.-K. Baik, Home gateway operating model using reference monitor for enhanced user comfort and privacy, IEEE Trans. Consum. Electron. 54 (2008) 494–500. doi:10.1109/TCE.2008.4560120.

[21] P. Busnel, P. El-Khoury, S. Giroux, K. Li, An XACML-based Security Pattern to achieve Socio-Technical Confidentiality in Smart Homes, J. Smart Home. 3 (2009) 17–26.

[22] A. Faravelon, S. Chollet, C. Verdier, A. Front, Enforcing privacy as access control in a pervasive context, in: 2012 IEEE Consum. Commun. Netw. Conf., IEEE, 2012: pp. 380–384. doi:10.1109/CCNC.2012.6181011.

[23] M. Jung, G. Kienesberger, W. Granzer, M. Unger, W. Kastner, Privacy enabled web service access control using SAML and XACML for home automation gateways, 2011 Int. Conf. Internet Technol. Secur. Trans. (2011) 584–591. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6148403 (accessed September 1, 2015).

[24] A. Müller, H. Kinkelin, S.K. Ghai, G. Carle, A secure service infrastructure for interconnecting future home networks based on DPWS and XACML, in: Proc. 2010 ACM SIGCOMM Work. Home Networks - HomeNets '10, ACM Press, New York, New York, USA, 2010: p. 31. doi:10.1145/1851307.1851315.

[25] T. Nixon, A. Regnier, V. Modi, D. Kemp, Web Services Dynamic Discovery (WS- Discovery), version 1.1, OASIS. (2009) 1–50. http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.pdf (accessed September 1, 2015).

[26] A. Banks, R. Gupta, OASIS Message Queuing Telemetry Transport (MQTT), version 3.1.1, OASIS. (2014) 1-81. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.pdf (accessed September 1, 2015).
[27] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, Web Services Description Language (WSDL) 1.1, Int. Bus. (2001) 1–40. http://www.w3.org/TR/wsdl (accessed September 1, 2015).
[28] K. Fysarakis, D. Mylonakis, C. Manifavas, I. Papaefstathiou, Node.DPWS: Efficient Web Services for the IoT, IEEE Softw. (2015). To appear.
[29] Mosquitto Open Source MQTT v3.1/v3.1.1 Broker. http://mosquitto.org (accessed September 1, 2015).
[30] K. Lawrence, C. Kaler, A. Nadalin, R. Monzilo, P. Hallam-Baker, Web Services Security: SOAP Message Security 1.1, OASIS Stand. Specif. (2006) 1–76. https://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf (accessed September 1, 2015).
[31] D. Whiting, R. Housley, N. Ferguson, Counter with CBC-MAC (CCM), (2003). http://tools.ietf.org/rfc/rfc3610.txt (accessed September 1, 2015).
[32] K. Rantos, K. Fysarakis, C. Manifavas, I.G. Askoxylakis, Policy-Controlled Authenticated Access to LLN-Connected Healthcare Resources, IEEE Syst. J. (2015) 1–11. doi:10.1109/JSYST.2015.2450313.

## BIOGRAPHIES

**Konstantinos Fysarakis** is a PhD candidate at the Department of Electronic & Computer Engineering of the Technical University of Crete. He received an MSc in Information Security from Royal Holloway, University of London and a BSc in Applied Mathematics from the University of Crete. His interests revolve around the security & dependability of embedded systems and the challenges of ubiquitous computing, publishing research papers and being involved in pertinent EU-funded projects. He is a member of the IEEE and also enjoys working on industry-related tasks, dealing with Information Security Management issues in particular (being an IRCA certified ISO 27001:2005 auditor).

**Othonas Soultatos** is a researcher at the Department of Electronic & Computer Engineering of the Technical University of Crete. He received a BSc in Computer Science from the University of Crete. His interests are in the area of lightweight communication protocols and their implementation using different programing environments, and also security in communication protocols and web-based applications.

**Dr. Charalampos Manifavas** has fifteen years of experience in the network and information security field, holding positions in the public and private sectors as a security engineer and consultant. He is an Assistant Professor at the Electrical Engineering and Computing Sciences Dept. of Rochester Institute of Technology Dubai, UAE. He holds a PhD in Computer Science from the University of Cambridge, an MSc in Communications Systems Engineering from the University of Kent and a BSc in Computer Science from the University of Crete. He has participated in several EU funded research projects and has published more than 40 research papers.

**Dr. Ioannis Papaefstathiou** is a professor at the Department of Electronic & Computer Engineering of the Technical University of Crete. He holds a PhD in Computer Science from the University of Cambridge, an MSc in Computer Science from Harvard University and a BSc in Computer Science from the University of Crete. He has been Principal Investigator in numerous competitively funded research projects in Europe and has published more than 80 papers in IEEE/ACM sponsored journals and premier international conferences. He is interested in the architecture and design of novel computer systems, concentrating on devices with highly constrained resources.

**Dr. Ioannis G. Askoxylakis** is a member of the Telecommunications and Networks Laboratory of FORTH-ICS. He holds a Diploma in Physics from the University of Crete, an MSc in Communication Engineering from the Technical University of Munich and a PhD in Engineering from the University of Bristol. His research interests lie in the fields of system and communication security, operational cybersecurity, computer security incident response and emergency response communications. He is the author of more than 40 publications in international conferences and journals, as well as book chapters, and has both organized and chaired several prestigious international conferences and workshops.