Designing Suitable Access Control for Web-Connected Smart Home Platforms

Sebastian Werner, Frank Pallas, and David Bermbach

Technische Universität Berlin
Information Systems Engineering Research Group,
{sw,fp,db}@ise.tu-berlin.de

Abstract. Access control in web-connected smart home platforms exhibits unique characteristics and challenges. In this paper, we therefore discuss suitable access control mechanisms specifically tailored to such platforms. Based on a set of relevant scenarios, we identify requirements and available technologies for fulfilling them. We then present our experiences gained from implementing access control meeting the identified requirements in OpenHAB, a widely used smart home platform.

Keywords: Access Control, IoT, Smart Home

1 Introduction

The Internet of Things (IoT) has been finding more and more adoption in the last few years and is seeing continuous growth. Gartner, for instance, expects 20.4 billion connected "things" to be in use by 2020 [10]. A particularly popular area in the IoT are the so-called smart homes where sensors and smart appliances are connected using often rule-based approaches to increase comfort for the home inhabitants.

Today, smart home platforms like OpenHAB[1] often run in isolated networks not connected to the public Internet. However, we expect this to gradually get replaced by more open deployments that include external services on the web and in the cloud, e.g., if-this-then-that (IFTTT)[2] or Amazon's Machine Learning service[3], through web APIs, e.g., for weather data, or even using devices such as Google Home[4] or Amazon Echo. For such deployments, however, access control is a crucial feature of the smart home platform both to protect the inhabitants' privacy but also to protect them from malicious attacks [8,14]. When the possibility of tampering with the physical world exists, security becomes even more important – some recent events underline this [2, 17].

While access control is obviously highly important in smart home platforms, there are a number of characteristics of such systems that make it hard or at least

[1] openhab.org
[2] ifttt.com
[3] aws.amazon.com/machine-learning
[4] madeby.google.com/home

non-optimal to simply reuse access control strategies from other domains, e.g., from file systems or mail servers. Therefore, we make the following contributions in this paper:

1. We identify smart home usage scenarios particularly relevant for access control and use those to derive unique requirements for access control in future smart home platforms.
2. We use OpenHAB, a widely used smart home platform, as a case study and first analyze its current access control approach. Building on these rather lacking techniques, we then describe our efforts of retrofitting OpenHAB with access control features that are ready for future deployments in open, web-based environments.
3. We identify key lessons learned for both users of current as well as developers of future smart home platforms.

This paper is structured as follows: We describe relevant usage scenarios in smart home environments and identify core requirements for access control in such systems in Section 2. Afterwards, in Section 3, we give a brief overview of the OpenHAB platform, describe its current access control approach and our efforts in making said platform ready for open, web-based deployments. Building on this, we discuss our observations and lessons learned in Section 4 before concluding in Section 5.

2 Access Control in Smart Home Systems

In matters of access control, IoT Systems and, in particular, smart home platforms differ significantly from other systems in various respects, leading to specific requirements that have to be met. While "basic use cases" of smart home platforms are often perceived trivial in matters of access control – all members of a family living in a house have full control, for example – things change significantly when "non-regular" use cases and scenarios come into play. Some such scenarios shall thus be briefly outlined in the following.

2.1 Relevant Usage Scenarios

One scenario that any smart home platform will sooner or later have to address properly comprises guests staying within the covered premises [13]. This may include private visitors as well as hotel- or AirBnB-like settings. In any such case, guests should, for example, be able to control and program the behavior of their windows' blinds as well as to monitor and control the temperature of their respective room during their stay but have no access to similar functions for other private rooms. Also, there might be a need to allow guests to read the status of shared devices like the temperature sensors in the hallway or to dynamically connect the smart home platform with external services (for receiving personalized email notifications or traffic forecasts, for example). Besides the core functionality required to implement all these aspects, associated processes

of user enrollment, provisioning of access rights, etc. should of course also be as uncomplicated and "frictionless" as possible.

Another common scenario refers to external trust persons legitimately sojourning in the covered area – e.g., cleaning or nursing staff or simply different friends of the inhabitants looking after plants. As such persons are commonly trusted to physically enter the premises and thus to have physical access to most devices located within (e.g., lamps, window blinds, heating, ...), it makes no sense to prevent them from having the same access in the digital domain. However, providing them with respective access on the basis of pre-known identities may often prove challenging: In the case of cleaning or nursing staff, for example, the actual persons legitimately present within the premises often differ unpredictably, depending on dynamic schedules and ad-hoc substitutions. Using their physical presence itself is therefore a more promising starting point for suitable access control mechanisms. Alternatively, an approach more in line with the process used for providing physical access itself – the cleaning company is trusted, thus provided with a key and authorized to hand over this key to its personnel – may also serve as blueprint for suitable access control here. A combination of several approaches may also be needed from time to time: for instance, staff may start out with limited privileges simply based on their employment relation and physical access to certain areas but may acquire additional rights over time based on increased trust at a personal level.

With increasing interconnection of things, services, and APIs, it also becomes more and more important to provide suitable access control to an open, specifically non-closed smart home platform. For instance, hitting a dash button to order some products also has monetary effects on the owner's bank account – controlling who can invoke external APIs should not be disregarded. Also, while external services can be integrated through invocation and polling, sometimes such third party services will have to trigger events in the smart home system. For instance, one might define a set of rules on IFTTT or FRED [5] that analyzes current wind measurements and controls window blinds to protect them from permanent damage. Or one could use an external service that offers precisely that functionality. Last but not least, there is a number of separate home automation gadgets, e.g., Google Home or Amazon Echo, or even simply a smartphone. Integrating such devices requires the necessary access control logic to grant extensive but not unlimited rights to those devices. All in all, this means that fine-grained access control with intelligent learning and delegation logic must not be limited to human users but also needs to extend to external services, platforms, and APIs.

Finally, the core problem that strict and highly elaborate access control models often prevent dynamic reactions to unforeseen situations will in all likelihood become increasingly prominent in the smart home context. Assuming a scenario where a person in need of care does not respond to calls on the door, it might be reasonable if, for example, police officers could request temporary access to an in-house camera or to the door lock from, e.g., the person's children living in another city. Alternatively, it might also be reasonable to allow certain parties

like emergency services to "override" access restrictions in a controlled manner that can be thoroughly examined afterwards. In any case, "preparing for the unanticipated" will be of utmost importance for suitable and usable access control in smart homes – not only in cases of emergency.

2.2 Technical Requirements and Mechanisms

Based on the above scenarios, we have identified a number of core requirements for access control in web-enabled smart home platforms. In this Section, we present these requirements and briefly discuss mechanisms we believe to be useful for addressing them. All requirements and mechanisms identified below are intended to be non-exclusive, allowing for multi-condition access control combining multiple criteria from the same category as well as from different ones.

Frictionless usage: Smart home scenarios pose a particular need for smoothness. Especially in the living context, smart technologies should not be perceived as separate "add-ons" but must rather be embedded or "woven into" [18] established behavioral patterns and habits. Guests – in a hotel as well as in a private context – should not feel bothered by a need to familiarize with smart components or by cumbersome setup or "enrollment" procedures. Necessary management efforts should also be low on the operator's side. This is particularly true for access control mechanisms, where repeated commissioning and decommissioning of users and fine-grained assignment of rights could easily raise significant overheads. For all our considerations below, ensuring frictionless usage is thus a core design goal. Furthermore, frictionless usage may also be supported by specifically targeted mechanisms of self-learning etc.[5]

Fine-grained access control: One core requirement that can be identified from the above scenarios is the need for fine-grained access control. It must be possible to provide different parties with different access rights for monitoring and control on a per-resource (devices, software components and services) level. To lower the efforts necessary for specifying and managing respective access rights, device grouping should also be possible. In terms of concrete mechanisms, this requirement resembles functionality already provided by most modern access control schemes. With minor adaptations, established mechanisms for fine-grained access control should thus be transferable to the smart home domain at reasonable effort.

Temporary permissions: As motivated by the guest-related scenario, temporariness of access rights plays a more prominent role in the smart home context than it typically does in other domains. Even though granted access rights might always be revoked manually, directly issuing them for a limited timeframe would

[5] Manifold further approaches could be thought of here. Kim et al. [13], for instance, also experimented with automated rights assignment based on social network and phone records analysis.

thus be of particular value for smart home platforms, also serving the goal of frictionless usage. Technology-wise, access tokens with limited validity periods are widely used in practice for comparable purposes. These might be used in combination with pre-existing user accounts (e.g., when access rights have been temporarily escalated and activities must be attributable to the respective user) or as a completely isolated modality of authentication, depending on the intended use case.

Context-driven permissions: The existence and relevance of physical context is one of the main specifics distinguishing access control in smart homes from other application domains. In the smart home domain, any device as well as any user striving for access has a physical presence and context which can serve as a foundation for access control concepts specific to smart home scenarios. In the above-mentioned case of cleaning or nursing staff legitimately being in a house, for example, any evidence proving that a user actually is within certain premises may be deemed a sufficient basis for granting access to all devices located in these premises [13]. Sufficiently reliable proofs for being in a certain room might, for example, be based on ultrasonic sound [7] or indoor light [11] used to transfer secrets to the client device. Independently from the concrete mechanisms used, however, we see permissions based on physical context – a particular form of attribute-based access control [16, 19] – as an important component of suitable access control for smart home and IoT scenarios.

Rights delegation: Scenarios like the one referring to cleaning or nursing staff above also motivate a need for rights delegation. Instead of access being granted based on physical access (which particularly proves impractical in the case that door locks are also managed by the access control system), rights delegation could also be used: The cleaning company would then be granted access to certain devices, combined with the possibility to "hand over" these access rights to its employees. Such capabilities for rights delegation may also serve the overall goal of frictionless usage and the need to "prepare for the unanticipated", as it allows to dynamically align access rights to changing needs. Rights delegation can be implemented in various ways, ranging from role delegation [1] to certificate driven solutions [4]. With a particular focus on IoT, rights delegation can be implemented through capability-based delegation [3], attribute-based access control, or through adaptations of the OAuth protocol [9].

On-request access provisioning by authorized parties: Closely coupled with the basic need for rights delegation is the requirement of being able to explicitly request access. As laid out in the scenario with a non-responding person and as obviously given in many less dramatic situations, actual access needs in smart home environments can hardly be foreseen completely but rather often emerge and need to be fulfilled ad-hoc. Beyond pure rights delegation, well-defined mechanisms for requesting not yet existing access from authorized persons and for respective "on-request delegation"[6] are thus an important building block of

[6] Elsewhere, this model is also called "ask for permission" [13].

"preparing for the unanticipated". In addition to those mechanisms already used for basic rights delegation, techniques for on-request delegation can build upon established concepts such as authentication proxies [15] or dynamic consent [12].

Break-glass overrides: Another approach of "preparing for the unanticipated" is the concept of break-glass access control. In this model, emergency access to resources (like, in the example above, the camera within a non-responding person's flat) is possible as soon as the "breaking" party (e.g., a police officer) can be reliably identified and appropriate notifications about the act of breaking are sent out to trigger ex-post evaluations. Having their roots in healthcare and disaster management, break-glass override mechanisms are highly relevant whenever strict access restrictions could potentially lead to significant harm in unanticipated situations. They might thus be of particular relevance in the smart home context whenever reasonable emergency access would today be achieved my means of physical destruction, for example. Technically, break-glass mechanisms can be implemented on top of or natively integrated into existing access control mechanisms [6], whereas the ability to actually perform an emergency access might be given to anybody or limited to a well-defined set of parties. Blockchain technology might be helpful for implementing non-disputable break-glass logs.

3 Case Study: OpenHAB

In the following, we discuss openHAB, the IoT framework we used to prototypically implement some of the mechanisms described in Section 2.2. We first give a brief overview of openHAB, its architecture and how it handles access control in its default configuration. Afterwards, we delineate how we extended its access control mechanisms to better meet the requirements we developed in Section 2.

3.1 System Overview

As a case study, we chose the OpenHAB project, mainly because it represents a typical IoT eco system and unlike most others is open source. OpenHAB is an extension of the Eclipse SmartHome project and explicitly built for easy integration of new devices, protocols, and services. Most of its components interact through a pub/sub event stream, the "**Open H**ome **A**utomation **B**us". The project also comes with a modern interface and one of the most exhaustive and fastest-growing repositories for device bindings and third party integrations on the market, which together led to a growing community of users.

The framework is based on Java and composed of a set of OSGi (Open Service Gateway Initiative) services and components. Its architecture can be broken down into three major component groups: 1) the core: including the message bus, configuration storage, logging; 2) binding provider: responsible for communicating with devices and services; 3) user interfaces and user services, including the rule engine employed to run user-defined code and the REST interface that can

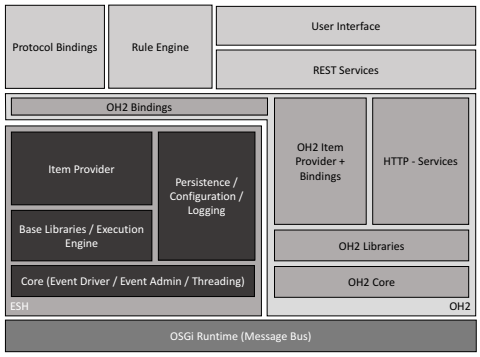

Fig. 1. Overview of the OpenHAB System Architecture. Showing the Eclise SmartHome Core (left) and the OpenHAB specific extensions (right) as well as common modules.

be utilized by external services. Figure 1 shows a more detailed view of the OpenHAB architecture.

Any extension to OpenHAB is added as an OSGi binding, which can access every other service within the OpenHAB runtime including all data stored within the system without any constraints.

3.2 Access Control in OpenHAB

OpenHAB has a very basic access control system, which only allows a single user/password combination to be set. If set, the respective user becomes the only entity able to access any and all of the openHAB features. By default, however, setting a user/password combination is neither enforced nor at least suggested. Having only one single account clearly limits openHAB's usability for more complex use cases as presented in Section 2.1.

Furthermore, OpenHAB enforces access control only in the third group of components (user interfaces), while bindings and user-defined rules can access any internal service or any other resource on the local or public network without providing any means of authentication. Even though clearly fostering the easy development and addition of new types of services and system resources as OpenHAB components, such a full-access paradigm is obviously unsuitable for more complex settings with hundreds or thousands of controlled devices, with the platform being interconnected with multiple external services, and with external users potentially being able to connect their own devices and external services: Under the current concept, just one exploitable or unexpectedly behaving component could suffice to compromise or disturb the whole installation. Together with the risk-prone dynamic update or addition of components inherent to the OSGi concept, this sums up to a security concept and subsystem that is everything but "production-ready" beyond rather trivial use cases in closed networks. Finally, none of the advanced requirements identified in Section 2.2 is currently fulfilled by OpenHAB.

3.3 Extending Access Control

Given OpenHAB's above-mentioned shortcomings, we set out to fix its core security model and to prototypically implement mechanisms for meeting the requirements identified in Section 2.2. For doing so, we had to overhaul most parts of OpenHAB's existing access control mechanisms and retrofit our own. To ease management effort and heighten practical usability in non-trivial use cases, we added a way to group devices in multiple overlapping security groups, established a way to record the use of each device, and introduced a rudimentary user registry.

In order to concentrate on the internal processes of the access control system, we confined our development efforts to an installation with only the REST interface being active. With our extensions, all access control information is transmitted using a mechanism based on JSON web token (JWT) in a custom HTTP-header. For use cases involving attribute-based access control, a client can either compute these tokens (based on context information, for example) or listen for a local broadcast token. Each response from our system can also contain a new token for the client to use, like a generated identifier that is similar to an automatically generated username.

Implementing fine-grained access control: Based on the just discussed infrastructure functionality, we implemented fine-grained access control by exploiting the already existing metadata infrastructure of OpenHAB. It provides a set of tags for each device; we use these tags to identify relationships between devices. Once an entity can prove access to a device, the user might also be granted access to other devices with similar tags. We use the groups and device capabilities encoded in these tags to assess the risk of unauthorized access. In the absence of other authentication information, the calculated risk is used to make access decisions. To fully support fine-grained access control, we also do not enforce access control simply at the user interface but at the core components responsible for issuing commands to devices.

Implementing rights delegation: We implemented rights delegation mainly by relying on the above-mentioned user management; it holds rights that each user accumulates over time. Delegation then is handled by linking the delegated sets of rights to a new user (thus following a user-driven approach to prevent uncontrolled onward delegation). The original rights holder can still use these rights, revoke or modify delegations, and delegate the same rights to further users.

Implementing temporary, context-driven, and on-request permissions: We used attributes codified in tokens to determine if a user could access a resource. Each attribute must also be signed, which allows the system to ensure that an attribute could not be manipulated. These attribute tokens can encode location information as well as context information. Once a request is issued to the system, these attributes are evaluated by the respective controller classes. All attributes

are evaluated individually for each device affected by the request. Depending on the evaluation outcome, it might be that the response only contains a filtered set of information. A controller might also decide only to perform a subset of the actions that were requested. Using similar concepts, we also implemented on-request access provisioning, where the controller puts respective commands on hold until a privileged user has approved it.

Implementing break-glass overrides: Finally, we added a break-glass override mechanism by allowing our access control system to be shutdown if the right set of keys is used. These keys are generated by a token generator that would have to be given to those parties that should be able to perform an override (the local fire department, a neighbor, etc.). The owner of the OpenHAB installation has total control over these generators and can disable them at any time. Once the break-glass override is invoked, all access control mechanisms are disabled, and only the audit component is still active. The notification system of the on-request delegation system is used to inform all privileged users what is happening and which token was used to invoke the breaking the glass override.

Means for frictionless use: To serve the overall goal of frictionless usage both on the user and the management side, our system can learn from each interaction and over time build a simplified model about what a specific user is allowed to do. Once multiple comparable requests (like "light X in room Y") have repeatedly been approved, the user no longer needs to ask for permission for similar requests. This is done by recording each request, each attribute used for that request and the outcome of these requests. We use this information to train a simple risk model. If the risk factor for a given request is low enough, then the request will be successful regardless of the content of other provided attributes. We also record these automatic approvals for a later audit.

Specific challenges of OpenHAB: Intuitively, implementing these features should have been comparably simple. However, OpenHAB's architecture made it rather difficult. In particular, the OSGi framework, which openHAB uses to dynamically load and replace processes at runtime, could have been used to disable any feature our access control add-on would provide. To get around this problem, we enhanced the control flow of information between all components responsible for executing commands (e. g., ItemProvider and Execution Engine). We did this by implementing an auditing framework that analyzed the stack trace of each incoming method call and only allows a predefined set of the classes to execute OpenHAB commands. Any OSGi service can still see all internal services, but if a non-listed service tries to create a request, all commands are dropped.

The OSGi framework also leads to another problem regarding the communication between our access control implementation and other components in the system. The original design of openHAB did not allow for session information to be transported to lower-level components. This meant that we could not filter communication between different layers on a per-resource basis. We, therefore, had to build a complicated workaround to transport simple session information

between classes: Given that we could not change methods inside the architecture without rewriting all OpenHAB device bindings, we had to transfer information indirectly. In particular, we created an OSGi service that interacted with the Java Debug Interface. This service allowed us to correlate session information with stack traces and method calls which allowed us to evaluate device interactions on an event by event basis. In matters of performance, maintainability, and transparency, this approach is, however, by no means ideal.

Besides these limitations, we were able to add the on-request-delegation system, location broadcasts and other extensions mentioned in Section 2.2 without any hindrance. Besides implementing the user interfaces for these services, we also added a set of cryptographic utilities that are, for instance, used to generate the time-dependent tokens using a HMAC-based one-time password algorithm.

4 Discussion

The lesson learned no.1 is that retrofitting access control is a rather bad idea; instead, it needs to be considered as a core design goal from the very beginning especially for an open, web-connected platform that integrates external services and data sources with a non-fixed user base. It is also important to handle access control not only on an exterior interface but rather internally on a component or device level. With OpenHAB's current overall system architecture, however, applying the presented techniques from Sect. 2.2 or even making the slightest extensions to the existing security design is virtually impossible even though we managed to achieve a working solution. However, considering that complexity is the worst enemy of security, the necessary workarounds and indirections for retrofitting advanced access control that we took cannot be recommended safely.

While we only analyzed OpenHAB in such detail, access control does not seem to be a core design goal in other platforms either. For addressing advanced scenarios as described in Sect. 2.1 and being web-ready including interaction with third party services, smart home platform developers should probably rethink their priorities. In this context, technology such as OSGi also has both benefits and disadvantages – deciding on such technology stacks should be the result of careful deliberation.

Finally, an important consideration is also to design access control in a way that is compatible with widely used, established web standards such as OAuth or even WS-Security. Otherwise, all integration of external services, data sources, and APIs is ripe for malicious exploitation and not advisable.

5 Conclusion and Future Work

The Internet of Things has been finding more and more adoption in the last few years and is seeing continuous growth; particularly, smart home technology has found more and more adoption. Smart home platforms, however, still mainly run in isolated networks not connected to the public Internet. We expect this to gradually change towards more open deployments including external services on

the web and in the cloud. In such deployments, however, access control based on mechanisms specifically tailored to the smart home domain is of utmost importance.

In this paper, we started by discussing unique use cases in future smart home platforms and used these to derive specific requirements for access control. Afterwards, as a case study, we analyzed OpenHAB (as a popular example of a smart home platform) in detail with regards to its access control mechanisms. Since we found these rather lacking, we then tried to retrofit state-of-the-art access control in compliance with our identified requirements. Through this case study, we learned that at least one of the current smart home platforms (and potentially even other more general IoT platforms) is not ready for future, more complex application scenarios. We also identified how smart home environments offer unique ways for access control based on physical context. Finally, we learned that retrofitting access control in existing smart home platforms is painfully complicated and cannot be recommended – both from a security and an effort perspective.

In our future work, we plan to test other IoT Hub platforms and see if some of the limitations we observed can be improved by using a more suitable platform where fewer indirections and workarounds are needed to integrate our fine-grained access control method. Furthermore, we intend to test our methods not only on edge devices but also expand our access control approach to the dispersed and interconnected fabric of fog computation.

Acknowledgments

This work partly been supported by the European Commission through the Horizon 2020 Research and Innovation program under contract 731945 (DITAS project).

References

1. Ahn, G.J., Mohan, B.: Secure information sharing using role-based delegation. Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on 2, 810–815 Vol.2 (2004)
2. Andy Greenberg: Hackers Remotely Kill a Jeep on the Highway—With Me in It, https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/
3. Anggorojati, B., Mahalle, P.N., Prasad, N.R., Prasad, R.: Capability-based access control delegation model on the federated IoT network. Wireless Personal Multimedia Communications (WPMC), 2012 15th International Symposium on pp. 604–608 (2012)
4. Aura, T.: Distributed access-rights management with delegation certificates. Secure Internet Programming pp. 211–235 (1999)
5. Blackstock, M., Lea, R.: Fred: A hosted data flow platform for the iot built using node-red. Proc. of MoTA 2016 (2016)
6. Brucker, A.D., Petritsch, H.: Extending access control models with break-glass. In: Proceedings of the 14th ACM symposium on access control models and technologies. pp. 197–206. ACM (2009)

7. Chen, K., Aljarrah, M., Bonnet, P.: Leveraging physical locality to integrate Smart appliances in non-residential buildings with ultrasound and Bluetooth Low energy. Proceedings - 2016 IEEE 1st International Conference on Internet-of-Things Design and Implementation, IoTDI 2016 1(iii), 199–209 (2016)
8. Dong, R., Ratliff, L.J.: Privacy in the Internet of Things. The Next Wave 21(2), 8–16 (2016)
9. Emerson, S., Choi, Y.K., Hwang, D.Y., Kim, K.S., Kim, K.H.: An OAuth based authentication mechanism for IoT networks. International Conference on ICT Convergence 2015: Innovations Toward the IoT, 5G, and Smart Media Era, ICTC 2015 pp. 1072–1074 (2015)
10. Gartner: Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016 (2017), http://www.gartner.com/newsroom/id/3598917
11. Grobe, L., Paraskevopoulos, A.: High-speed visible light communication systems. Communications ... 1(December), 60–66 (2013)
12. Kaye, J., Whitley, E.A., Lund, D., Morrison, M., Teare, H., Melham, K.: Dynamic consent: a patient interface for twenty-first century research networks. European Journal of Human Genetics 23(2), 141–146 (2015)
13. Kim, T.H.J., Bauer, L., Newsome, J., Perrig, A., Walker, J.: Access right assignment mechanisms for secure home networks. Journal of Communications and Networks 13(2), 175–186 (2011)
14. Liu, J., Xiao, Y., Chen, C.P.: Authentication and access control in the Internet of things. Proceedings of the 32nd IEEE International Conference on Distributed Computing Systems Workshops 2012 pp. 588–592 (2012)
15. Mayrhofer, R.: A context authentication proxy for IPSec using spatial reference. In: Proc. TwUC 2006: 1st International Workshop on Trustworthy Ubiquitous Computing. p. 449–462. Austrian Computer Society (OCG), Austrian Computer Society (OCG) (December 2006)
16. Sicari, S., Rizzardi, A., Grieco, L.A., Coen-Porisini, A.: Security, privacy and trust in Internet of Things: The road ahead. Computer Networks 76, 146–164 (2015)
17. Vallance, C.: Car hack uses digital-radio broadcasts to seize control (2015), http://www.bbc.com/news/technology-33622298 http://www.bbc.co.uk/news/technology-33622298
18. Weiser, M.: The computer for the twenty-first century. Scientific American pp. 94–100 (1991)
19. Yuan, E., Tong, J.: Attributed Based Access Control (ABAC) for web services. Proceedings - 2005 IEEE International Conference on Web Services, ICWS 2005 2005, 561–569 (2005)