An IoT-enabled Framework for Context-aware Role-based Access Control

Giorgio Delzanno, Giovanna Guerrini

DIBRIS, University of Genova, Italy

Abstract. We present a framework for enforcing the application of context-aware Role-based Access Control policies based on an Internet of Things eco-system inspired by the Google's Physical Web. In this setting we are interested in capturing three contextual dimensions, namely who-where-when, and using these information to restrict access to shared resources. Formally, the framework consists of features types, an automata-based model of time-sensitive roles, context-aware permission rules, and an IoT infrastructure based on Eddystone Beacons for validating a policy against the current state of users.

1 Introduction

In modern distributed systems based on the Internet of Things (IoT), security layers of software applications must be tightly integrated with the underlying system/network infrastructure. The integration is needed in order to increase the privacy of user data, to ensure availability of access control configuration services, and to integrate edge and fog components in the, possibly seamless, eco-system that governs information assurance. In particular, devices at the edges of an IoT system can be employed as active parts of location-based and time-dependent access control policies enforcement mechanisms. Furthermore, a tighter integration between the security layer and distributed infrastructures could provide better provenance on data and information flows.

According to this idea, in this paper we present a framework for enforcing the application of geo-referenced Role Based Access Control (RBAC) policies [5] based on an IoT infrastructure inspired by the Google's Physical Web paradigm [4,6,7,8,9,11,12,13,14,15]. The Google's Physical Web project enables smartphone users to interact with physical objects and locations through the use of beacon technology. Beacons are low cost radio transmitters that typically transmit a unique ID on a regular interval, e.g., 100-1000ms, in a range of approximatively 30 meters. Bluetooth-enabled devices can detect a beacon and receive its corresponding identifier, following the so-called lighthouse metaphor. Smartphone applications can use such IDs to signal their physical presence in

SEBD 2018, June 24-27, 2018, Castellaneta Marina, Italy.

the beacon vicinity to a remote server and the limited transmission range of these transmitters provides precise users localization. The Physical Web is particularly useful for indoor scenarios equipped with low cost devices such as beacon transmitters. This technology can be applied to set up non-invasive and precise localization methods, with easily configurable privacy protection layers (e.g., associations between user IMEI's and beacon identifiers, etc), to generate location-based provenance meta-data, and to ensure a tight integration between access control policies, physical spaces, and policy enforcement mechanisms.

Our framework consists of features types, context-aware permission rules, and an IoT infrastructure for validating a policy against the current state of users. In this setting, we are interested in capturing three contextual dimensions, namely who-where-when, and use these information to restrict access to shared resources. Users are assigned to stateful roles. A role must be considered as a slowly changing condition of users, whereas a state represents time-dependent a behavior associated with a specific role (e.g. typically a faculty member can act as instructor, mentor, course attendee, etc). Similarly, locations have time-dependent states that capture possible different usages of the same physical space (e.g. a room can be used for meetings, seminars, courses, etc). Our time model is based on deterministic finite-state automata inspired by logical specification languages for the linear time flow. Access control rules are defined on top of role and location states. Therefore, they can be used to define complex (i.e. automata-based) time-dependent access policies.

We illustrate the main ideas underlying the specification language of context-aware control access policies using an example. Consider a campus which physical spaces are organized in buildings, floors, and rooms. A room may have several usages, e.g., meeting room, seminar room, lecture room. The usage depends on a predefined time schedule. The same room can have different usages in different time slots. Users can assume different roles, e.g., student, teacher, mentor. Furthermore, users perform actions like consulting and updating records related to attendance at a given lecture, finding information on teachers, retrieving statistics, etc. Permission rules formalize access control policies to public and private data. Permissions are context-aware in that access to specific data can be constrained by the physical location of the user. Permission rules have to specify properties such as: a student can update its attendance record only when she/he gets access to the system from a classroom, teachers can get statistics on student attendance anywhere, students can get information of teacher presence anywhere in the department building, etc. This scenario can be nicely formalized by introducing a time-dependent stateful model for roles and locations. The timed model is based on a linear flow of time represented as an automata with a total transition function (i.e., automata that have a lasso shape). Roles and locations have an associated labeling function dependent of time that determines the dynamics in each role specialization and in the usage of physical spaces. Validation is defined against permission rules associated to user operations by considering a notion of state based on the current user location and the system time.

The main novelty of our proposal w.r.t. classical approaches like GeoRBAC [5], TRBAC [1] or proximity frameworks like PROX [10] is the combination of an automata-based specification of context-aware access control model with an IoT validation infrastructure that can be automatically reconfigured by encoding a policy into a relational data model and a corresponding control program for policy enforcement.

A prototype of the system is currently under evaluation in our University with the aim of providing an automated support for the management of typical Academic Campus services like registration of attendance to lectures and meetings, rooms and lectures allocations, etc. Specifically, lecture attendance is often a problem for the performance of academic degrees (especially bachelor). Intermediate tests are often used as a way to encourage students to actively attend lectures and take exams. The platform provides (physical) web services and mobile applications for the registration and analysis of lecture/meeting attendance. Beacons are used to detect, in almost real-time, the presence of a student in a lecture room. Our specification language can then be used to formally configure access control to private data ensuring that users can modify their own record only when they are physically present in a given room authenticated in the system with their own smartphones. Furthermore, the same architecture can be applied in every scenario that requires a verified list of attendees, for example in the case of meetings in which it is mandatory to generate on-the-fly reports or collect results of several voting during the same meeting.

A prototype of the system has been developed using the Node.js server-side technology and Android for dedicated mobile app interfaces. The internal structure of Node.js allows the system to handle a large number of connections in a short period of time reducing potential risks of denial of service failures and attacks. Furthermore, Android OS provides natively support for beacons, for secure network connectivity, and for protecting device resources. In our system private data are exposed to the server only after obtaining permissions from the users. The system has been tested in a real scenario in the Database course of the Computer Science Bachelor degree at the University of Genova (more than 120 students).

In the paper, we present the specification language, the validation mechanism, system requirements, design principles, and implementation choices of the proposed architecture. The remainder of the paper is organized as follows. Sections 2 and 3 specify the model, in terms of policy specification language and enforcement. Section 4 focuses on infrastructure and system requirements. Section 5 discusses software related issues (architecture, data model, and prototype solutions) while Section 6 concludes.

2 Time Flow, Roles, Locations and Operations

Based on models for georeferenced and temporal access control languages [1,2,3,5], in our model both roles and locations are assigned a time-dependent notion of state. First of all, our notion of context is based on three dimensions: who, where,

and when. In other words we assume that the IoT infrastructure can provide precise information on current location and role (state) of each user in the window with granularity associated to the time model selected in the specific policy. To formalize these ideas, we first define our model of time.

Linear Time Model In this paper we consider a linear time model defined via a finite automata $A_T = \langle P, succ \rangle$, such that P is a finite set of time points and $succ : P \to P$ is a total function that defines the successor $succ(p)$ of any point $p \in P$. In other words, A_T is an automata that can only consist of an initial finite suffix entering into a finite and simple loop. The automata A_T will be used to model different time granularities, e.g., possibly repeated time/days slots during a week, months, etc.

Roles We define roles using feature types, i.e., a partially ordered set of entities. More specifically, the set of roles is defined by a finite set $Role$ equipped with a binary relation $\sqsubseteq$ such that $\langle Role, \sqsubseteq \rangle$ forms a lattice, i.e., $Role$ has top and bottom element, and every subset of elements in $Role$ admits a least upper bound and a greatest lower bound. The $\sqsubseteq$ relation can be viewed as a hierarchy relation (specialization) that can be used to build role hierarchies. For instance, $bachelorstudent \sqsubseteq student$ Every role $r \in Role$ comes with a finite set of possible states $State_r$ and a labeling function $\rho_r : P \to State_r$ that associates a given state to every time point. The labeling function must be compatible with $\sqsubseteq$ in that if $R_1 \sqsubseteq R_2$ the R_1 inherits all permits of R_2. In our model permits are associated with states, thus compatibility requires that if $R_1 \sqsubseteq R_2$, then $State_{R_2} \subseteq State_{R_1}$, i.e., state containment is contravariant w.r.t. role specialization.

Locations The set of physical spaces $Location$ is a finite set equipped with a binary relation $\sqsubseteq$ such that $\langle Location, \sqsubseteq \rangle$ forms a lattice. Every $l \in Location$ comes with a finite set of possible states $State_l$ and a labeling function $\rho_l : P \to State_l$ that associates a given state with every time point. Similarly to roles, the location labeling function must be compatible with the hierarchy relation, i.e., if $L_1 \sqsubseteq L_2$, then $State_{L_2} \subseteq State_{L_1}$, i.e., state containment is contravariant w.r.t. location specificity (i.e. specific users may have enlarged permissions in fixed locations inside a building).

Operations Operations are defined by a finite set of labels Ops.

Example 1. As an example, consider a linear time automata A_T with states $\{0, 1, 2, 3, 4\}$ s.t. $succ(i) = i + 1 \ mod \ 5$ s.t. $1 = Mon$, $2 = Tue$, etc. Now let us consider the set

$$Role = \{Student, Teacher\}$$
$$State_{student} = \{Attendant, Mentor\}$$
$$State_{teacher} = \{Teacher\}$$

where we omit top and bottom elements for simplicity. We assume here that users can be either students or teachers that access with two types of resources. Now we assume that $\rho_{Student}(i) = Attendant$ for $i \in \{0, \ldots, 3\}$, and $\rho_{Student}(4) = Mentor$. Furthermore,

$$Location = \{Building, Floor, Room1, Room2\}$$

s.t. $Room \sqsubseteq Floor \sqsubseteq Building$,

$$State_{Room1} = State_{Room2} = \{Meeting, Course, Floor, Building\}$$
$$State_{Floor} = \{Floor, Building\}$$
$$State_{Building} = \{Building\}$$

where we omit top and bottom elements for simplicity. Furthermore, $\rho_{Room1}(i) = Course$ for $i \in \{0, \ldots, 3\}$, and $\rho_{Room1}(4) = Meeting$, $\rho_{Room2}(i) = Meeting$ for $i \in \{0, \ldots, 4\}$.

Finally, we consider the following set of operations

$$Ops = \{GetRecord, UpdateRecord, FindTeacher, GetStatistics\}$$

The get/update record actions correspond to read/write operations on database containing student activity reports, such as course attendance data, mentoring activities, etc, that require certified time accountability, e.g., courses in which attendance is mandatory for a given percentage of hours. FindTeacher allows users to check for the presence of a teacher during specific time window and in specific locations. GetStatistics corresponds to read operations on aggregate data such as course attendance.

3 Access Control Rules and Policy Enforcement

Based on the previous sections, we are now ready to introduce our notion of permissions in order to define context-aware access control rules.

Definition 1 (Permission). *A permission rule is a tuple*

$$p = \langle Op, RoleState, LocationState \rangle$$

where $Op \in Ops$*,* $RoleState$ *is a role state in* $\bigcup_{r \in Role} State_r$*, and* $LocationState$ *is in* $\bigcup_{l \in Location} State_l$*.*

To clarify the meaning of permission rules, we need to introduce users, sessions, and then a validation mechanism.

Let Id be a denumerable set of user identifiers. Every user has an associated labeling function *session* that associates to the user an instance role. Permissions are then defined by rules associated to role and location states. More precisely, we introduce a notion of current state of a user via a state function defined as follows. For $u \in Id$, $state(u, t) = loc$, where we assume that loc is an instance of $Locations$.

Definition 2 (Validation of an operation). *User u can perform operation Op at time t if and only if there exists a permission rule $p = \langle Op, RS, LS \rangle$ s.t.*

- $state(u, t) = loc$,
- $session(u) = role$,
- *for every* $R \in Role$ *s.t.* $role \in R$ *we have that* $\rho_R(t) = RS$
- *for every* $L \in Location$ *s.t.* $loc \in L$ *we have that* $\rho_L(t) = LS$.

Example 2. We list below some examples of permissions.

- $p_1 = \langle UpdateRecord, Attendant, Course \rangle$ s.t. students can update their attendance records from the room of a course only.
- $p_2 = \langle UpdateRecord, Mentor, Meeting \rangle$ s.t. mentors can update their mentor accounting records from a meeting room only.
- $p_3 = \langle GetStatistics, Teacher, Building \rangle$: teachers can consult the student records anywhere in a building.
- $p_4 = \langle FindTeacher, Mentor, Building \rangle$: only mentors are allowed to use the service to locate a teacher in a building.

4 IoT Infrastructure

A prototype of the context-aware access control validation mechanism has been implemented using a IoT infrastructure based on the combination of beacons, wi-fi network technology, mobile, web, and database management components a typical scenario of the Physical Web paradigm proposed by Google. Our operative scenario is a typical example of an Academic Campus and we assume that each classroom is equipped with at least one beacon configured with a unique identifier consisting of three subfields called UUID, MajorID, and MinorID. In our experimental setup we adopted Estimote beacons. For lecture rooms for at most 180 students a fix beacon turned out to be sufficient to detect all enabled smartphones. We also assume that rooms of staff members are equipped with beacons to inform students of their presences in the building and office-hour updates. Finally, we assume that each room provides wi-fi access to students, e.g., via Eduroam (`https://www.eduroam.org/`) access points. A (Web) server must be installed either on an external cloud provider or on a local server. In both cases it must be reachable from the local network, i.e., the firewall configuration must provide access to Web APIs needed by the smartphone and Web applications.

The platform is based on a combination of hardware and software solutions that must be deployed in physical spaces like lecture and meeting rooms in an academic campus. The infrastructure and system requirements can be summarized as follows.

1. The system should provide *precise indoor localization* of users via their smartphones. The required precision must be in the order of a few tens of meters (i.e., to cover a lecture room).

2. *Registration* of user presence must be done in *real-time*. The server must be reachable from every lecture room.
3. *Rooms* must be viewed as *geofences*. Only users inside a given room should be able to update data with georeferenced permissions.
4. The system should be *resistant to server failures*, e.g., by using multiple server instances on different machines.
5. Logged *data must be stored persistently* in a data storage system. Data storage must be accessible to the application and administrators, only.
6. *Users must be informed* that the server makes use of localization data.
7. *Users must give their permission* for releasing data stored in the device (IMEI).
8. *Data* stored in the server *should not be released to third parties*.
9. Users must be *uniquely identified via official credentials* such as the matriculation (staff) number assigned by the central administration.
10. User devices must be *identified uniquely using IMEIs*. Every user must associate a unique device (IMEI) to the corresponding personal identifier.
11. *Data* exchanged with the users must be *encrypted and sent on secure channels*.
12. To *limit the budget*, hardware and software infrastructures must be composed by *low cost devices*, standard networking services provided in academic environments (e.g. server on virtual machines), open-source software and development frameworks for server-side and mobile applications.

To meet all the above requirements, we need to satisfy constraints both at the physical level (infrastructure and hardware) and at the software level (networking and platform).

5 Software Platform Architecture

The software platform consists of an app with different views depending from the current user (e.g., students, teacher, administrator), a Web application, and a persistency server as described in the next sections. The server provides secure APIs (secure TCP) for accepting user requests and for sending notifications. Persistent data are stored in a MySQL server accessible only from the server. The Web application allows staff members to create and modify user profiles, visualize historical data and statistics for both classrooms and individual students. Although responsive, this functionality must be viewed as an access point designed for a traditional (non mobile) browser. *Role* and *Location* are modeled using relational schemes. The state machines that defines *RoleStates* and *LocationStates* is stored as a relation containing a key for the time points, and states for each role instance. Thus validation of user state can be reduced to a query in the record associated to the current time point. Machine time is mapped to virtual time used in the access control specification by using a mapping function that is selected by the administrator. The mapping models the granularity of the considered time model (e.g, it extracts current day and hour from machine

Matricula	IMEI
3471890	980000832471652
3471891	990000551621881
3471895	990000144425624

Table 1. Association between student ID and IMEI

	9-11	11-13
Mon	CS1 (room 1)	Seminar (room 1)
Tue	CS2 (room 1)	Seminar (room 2)
Wed	CS1 (room 1)	Seminar (room 2)
Thu	CS2 (room 1)	Seminar (room 2)
Fri	CS3 (room 1)	Tutoring (room 2)

Table 2. Weekly Room Allocation Plan

time or sim.). This provides an alignment between machine time and automata time used in the enforcement step.

As an example, assume that Alice Smith is enrolled in the first year of a Computer Science Bachelor degree. Alice has student ID 3471890 and the association between her identifier and the IMEI of her smartphone (and of other two students) is shown in Table 1. The association between beacons identifiers and rooms is shown in Table 3. According the permission rule $p_1 = \langle UpdateRecord, Attendant, Course\rangle$ and $p_2 = \langle UpdateRecord, Mentor, Meeting\rangle$ attendance and tutoring hours registration for Alice Smith is synchronized in accordance to the data in Table 2 and Table 3 and the current location of Alice and time. Timing is based on the server time in order to avoid manipulation of timestamps sent with user requests. Registration in all other cases (different location, times) is forbidden.

The middleware underlying the platform has been implemented combining different technologies. The server has been implemented using the Node.js IoT framework, an efficient server-side development framework based on Javascript and on the npm eco-system. Node.js provides very efficient packages for handling secure TCP connections, Web servers, and applications. Node.js server-side libraries are optimized for network intensive applications even when executed on single host or cluster. Indeed, Node.js allows to handle a large number of connections in a short period of time reducing potential risks of denial of service (this is very useful in our context since the system has to handle hundreds of requests in a few minutes). This property is due to the internal structure of the Node.js event-driven engine. Requests are not handled using multiple threads as in the Apache server model since the Node.js engine is based on an event loop that executes callbacks sequentially. Callbacks are picked from multiple priorities FIFO queues. Furthermore, thread pool implemented using the C++ libuv concurrency library supports the execution of asynchronous callbacks. Differently from server architectures based on multiple threads, in Node.js the response to connection requests require few system resources since they basically require

Physical space	beaconID
room 1	101
room 2	102

Table 3. Association between rooms and beacons

the emission of events whose synchronous effect is that of enqueuing callback invocations in the I/O queue. This choice mitigates the risk of classical denial of service attacks based on a high number of simultaneous requests that could congest the server by exhausting system resources (e.g., creation of new threads to scale up the server).

The prototypes of the mobile applications have been developed in the Android OS. Android OS provides native support for beacons (e.g., using OS notification management or beacon SDKs like Estimote SDK). It also provides secure network connections and access control policy management that protects private data. Private data are exposed to the server only after obtaining permissions from the owner. The authentication mechanism built on top of associations between user and device identifier relies on secure network connections (secure TCP sockets) between the smartphone application and the server. A smartphone app, configured to detect beacons using libraries like the Estimote SDK, is provided in two different versions: students and teachers. The *student app* provides sign-up, sign-in, and sign-out and a wide range of functionalities. The sign-up service associates the smartphone IMEI to the unique student enrollment number and to his contact details. After the first user registration, sign-in is automatically enabled whenever the app is activated by the user. Indeed, the server can retrieve the IMEI of the smartphone from the initial connection request issued by the app. Thus, the silent authentication stage is based on the association between the user identifier and the registered IMEI. Upon authentication, students can register their attendance to a given lecture in a given time-slot. The lectures schedule of each student is indeed synchronized, server-side, with the lectures and rooms allocations plan, for each enrollment year. Thanks to this synchronization, attendance is enabled only in specific time-slots and in physical spaces. In addition the app provides user interfaces for visualizing the historical data stored in the persistence server (profile, attendance log) and interfaces to visualize, via a calendar widget, lecture plans (according to the corresponding study plan), seminars, and other events registered by staff members.

Using protocols similar to those adopted for the student view, the *teacher app* provides a user interface for visualizing statistics on students attendance, lectures and events calendar. The aggregated number of presences provides indications for each course trend: it is indeed possible to understand if the number of students attending at the beginning of the course decreases significantly or remains stable during the semester. It is also possible to check if there are days or time-slots with a significant decrease in attendance and consequently try to implement strategies to avoid such drop out. The app also provides a control widget for the notification of presence in office-hours. Staff members can create, modify, and

delete events that will be notified to users and visualized in their calendar view. Furthermore, they can access the room allocation service that is moderated by a dedicated administrator.

6 Conclusions and Future Work

The Google's Physical Web project – center stage at the Google IO developer conference in 2016– was conceived to enable smartphone users to interact with physical objects and locations through the use of beacon technology. The Google Physical Web architecture has been supported with physical devices such as Estimote beacons and client API's. Beacons are low cost radio transmitters that typically transmit a unique ID on a regular interval, e.g. 100-1000ms, in a range of approximatively 30 meters. Bluetooth-enabled devices can detect a beacon and receive its corresponding identifier, following the so-called lighthouse metaphor. Smartphone applications can use such ID to signal their physical presence in the beacon vicinity to a remote server and the limited transmission range of these transmitters provides precise users localization. The typical application domain of physical web is that of proximity marketing [8]. In this context, beacons are located nearby specific products and smartphone applications, enabled to detect beacons, redirect the user towards web sites with details on the products, brand, coupons, special offers, etc. Beacons have been applied in other domains like indoor localization [6,7,9,11,15], crowdsensing in public transportation [4], tourism [13], usage of physical spaces [12]. The main novelty of our physical web application comes from the use of beacons as enabling-technology to enforce context-aware access control policies. As a future work, we plan to extend both the specification language and the compilation of permission rules into an enforcement eco-system (devices, apps, data model, control program) and to study more applicative scenarios both in academic environment as well as in the enterprise domain.

Acknowledgments The authors would like to thank Claudio Orlando for his contribution to the prototype implementation.

References

1. E. Bertino, P. A.. Bonatti, and E. Ferrari. TRBAC: A temporal role-based access control model. *ACM Trans. Inf. Syst. Secur.*, 4(3):191–233, 2001.
2. E. Bertino, B. Catania, E. Ferrari, and P. Perlasca. A logical framework for reasoning about access control models. *ACM Trans. Inf. Syst. Secur.*, 6(1):71–127, 2003.
3. E. Bertino and E. Ferrari. Information security. In *The Practical Handbook of Internet Computing.* 2004.
4. D. Cianciulli, G. Canfora, and E. Zimeo. Beacon-based context-aware architecture for crowd sensing public transportation scheduling and user habits. In *The 8th International Conference on Ambient Systems, Networks and Technologies (ANT 2017) / The 7th International Conference on Sustainable Energy Information Technology (SEIT 2017)*, pages 1110–1115, 2017.

5. M. L. Damiani, E. Bertino, B. Catania, and P. Perlasca. GEO-RBAC: A spatially aware RBAC. *ACM Trans. Inf. Syst. Secur.*, 10(1):2, 2007.
6. W. He, P.-H. Ho, and J. Tapolcai. Beacon deployment for unambiguous positioning. *IEEE Internet of Things Journal*, 4(5):1370–1379, 2017.
7. J.-H. Huh and K. Seo. An indoor location-based control system using bluetooth beacons for IoT systems. *Sensors*, 17(12):2917, 2017.
8. K. Eun Jeon, J. She, P. Soonsawad, and P. Chet Ng. BLE beacons for internet of things applications: Survey, challenges, and opportunities. *IEEE Internet of Things Journal*, 5(2):811–828, 2018.
9. T. Kaulich, T. Heine, and A. Kirsch. Indoor localisation with beacons for a user-friendly mobile tour guide. *KI*, 31(3):239–248, 2017.
10. M. S. Kirkpatrick, M. L. Damiani, and E. Bertino. Prox-RBAC: A proximity-based spatially aware RBAC. In *19th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2011*, pages 339–348, 2011.
11. A. Mackey and P. Spachos. Performance evaluation of beacons for indoor localization in smart buildings. In *2017 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2017*, pages 823–827, 2017.
12. R. Purta and A.n Striegel. Estimating dining hall usage using bluetooth low energy beacons. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing and ACM International Symposium on Wearable Computers, UbiComp/ISWC 2017*, pages 518–523, 2017.
13. G. Sato, G. Hirakawa, and Y. Shibata. Push typed tourist information system based on beacon and augumented reality technologies. In *31st IEEE International Conference on Advanced Information Networking and Applications, AINA 2017*, pages 298–303, 2017.
14. A. Weisling and A. Xambó. Beacon: Exploring physicality in digital performance. In *Proceedings of the Twelfth International Conference on Tangible, Embedded, and Embodied Interaction, TEI 2018*, pages 586–591, 2018.
15. J. Zhu, K. Zeng, K.-H. Kim, and P. Mohapatra. Improving crowd-sourced wi-fi localization systems using bluetooth beacons. In *9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON 2012* , pages 290–298, 2012.