A Generic Framework for Context-Based Distributed Authorizations

Ghita Kouadri Mostéfaoui[1] and Patrick Brézillon[2]

[1] Software Engineering Group
Department of Informatics - University of Fribourg
Rue Faucigny 2, CH-1700
Fribourg, Switzerland
Ghita.KouadriMostefaoui@unifr.ch
[2] LIP6, case 169
Université Paris 6
Rue du Capitaine Scott 8, 75015
Paris, France
Patrick.Brezillon@lip6.fr

Abstract. In conventional security systems, protected resources such as documents, hardware devices and software applications follow an On/Off access policy. On, allows to grant access and off for denying access. This access policy is principally based on the user's identity and is static over time. As applications become more pervasive, security policies must become more flexible in order to respond to these highly dynamic computing environments. That is, security infrastructures will need to be sensitive to context. In order to meet these requirements, we propose a conceptual model for context-based authorizations tuning. This model offers a fine-grained control over access on protected resources, based on a set of user's and environment state and information.

1 Introduction

Research in the security field covers many aspects such as the improvement of cryptographic algorithms in order to make them more resistant to hackers, implementation of new authentication methods and designing access control mechanisms, etc. In traditional security systems, the security policy is pre-configured to a static behavior and cannot be seamlessly adapted dynamically to new constraints. This situation is due to the lack of consideration for the context in existing security systems. As a consequence, there is a lack of clearly defined conceptual models of context and system software architectures.

The goal of our research is to develop a conceptual framework for context-based security systems. Context-based security aims at adapting the security policy depending on a set of relevant information collected from the dynamic environment. As the environment evolves, the context change, some contextual elements being integrated in the proceduralized context, others leaving the proceduralized context [1]. Thus, security policies dynamically change in order to cope with new requirements.

P. Blackburn et al. (Eds.): CONTEXT 2003, LNAI 2680, pp. 204–217, 2003.

Our model is intended to handle all the components of a security system including authentication, privacy and authorization and may be easily extensible to include new security modules. However, this article discusses essentially resources access control or authorizations in the case of distributed systems, where a set of independent computers and devices communicate via a network in order to share data and services.

The structure of the paper is as follows. Section 2 discusses security issues in ubiquitous applications. The main contributions of the paper are presented in Section 3. The foundations for designing context-based authorizations frameworks are laid in Section 4. Section 5 concludes this paper with an outline of future research directions.

2 Security Issues in Ubiquitous Applications

The use of widely distributed resources provides a huge potential for expanding the way that people and businesses communicate and share data, provide services to clients and process information to increase their efficiency. This broad access has also brought with it new security vulnerabilities. Security systems developed now suppose a given and static framework, when attacks generally try to bypass these static contexts of effectiveness of security systems. Amazingly, security has often been the last requirement in designing such dynamic environments. This situation is due to the high cost of security infrastructures, export controls of cryptography technologies and the lack of experts in the security field for specific applications [2]. This is particularly true inside corporate networks where a firewall is assumed to keep all hackers out [3]. Firewalls are, however, not sufficient to protect shared resources. The main function of a firewall is to block unwanted traffic and hide vulnerable internal-network systems. It provides no data integrity and does not check traffic not sent through it, which means that it cannot protect the corporate network from internal attacks. People inside the network may maliciously or unintentionally reveal critical data to unauthorized users or disturb the well-working of the system. As a conclusion, firewalls should always be viewed as a supplement to a strong security policy.

According to Merriam-Webster a policy is *"a definite course or method of action selected from among alternatives and in light of given conditions to guide and determine present and future decisions."*

In the same spirit, we define a security policy as *"a set of rules that monitor all the security components behaviors acting on the framework to secure."* The security policy must be concise, descriptive and easily implementable. Security components consist of access control lists, cryptographic algorithms, and users authentication tools. They act over the following security levels: network and application levels.

2.1 The Network Level

Networks are all about the sharing of data and applications. In recent years, network security breeches have increased in occurrences and more importantly, in

severity. Security in these environments is thus a perquisite. Actually, there is no single technique to ensure reliable networks but different technologies (firewalls, encryption, etc. . .) are combined together in order to face security attacks. They try to extend security frameworks by combining them, but staying in a static approach. The next generation of Internet protocol (IPv6) is intended to add new security features at the network level over its predecessor version.

2.2 The Application Level

Network-based security suffers from some limitations on the kind of security checks that can be performed. The reason is that network-based security systems do not operate on a high level of data abstraction and cannot interpret the content of the traffic. They only know about hosts, addresses, and network related concepts. Application-based security is in contrast intended to provide a security layer based on user roles and identity along with other high level concepts such as protected resources and access policies. Our context-based security model is intended to operate at the application level. This does not mean that there is no need for network-level security. The main reason is simply that it is much cheaper to reconfigure the security infrastructure at the application level than at low-level (network).

Other requirements must be taken into account regarding the security policy. Following the definition given above, the security policy is intended to manage all the security components of the distributed system. Namely, the authentication, authorization, integrity and confidentiality modules and must be easily extensible to manage newly integrated modules. Following the aim of our work, the security policy must also be ***reconfigurable*** depending on the user and application environment context. This leads to the definition of a ***context-based*** security policy. Due to the pervasive nature of recent distributed environments, an additional requirement is the definition of ***shared*** policies. These features will be detailed in the following section.

3 Research Aim and Scope

Works addressing security issues in pervasive computing, basically provide technical solutions such as authentication, access control, integrity and confidentiality, and the security models are generally static. That means that they are built according to already identified threats. The resulting infrastructures are thus, very difficult to adapt to new threats. This work, rather, focuses towards a new aspect of security. We believe that more secure systems can be achieved by adding to these systems the ability to automatically adapt their security policy depending on new constraints. These constraints are dictated by the user's and application environment. Figure 1 illustrates this idea. The distributed application is controlled with an initial security policy in an initial context. This context is continually changing in request to triggers (dynamic changes in the environment). The security policy must then adapt itself to the new context.

Our approach will thus combine the two fields of context-aware computing and security in pervasive computing in order to provide the foundations for "context-based security".

3.1 Related Work

Integrating security with context-aware environments is a recent research direction. Most of the efforts are directed to securing context-aware applications. In [4] and [5], Covington et al. explore new access control models and security policies to secure both information and resources in an intelligent home-environment. Their framework makes use of environment roles [6]. In the same direction, Masone designed and implemented RDL (Role-Definition Language), a simple programming language to describe roles in terms of context information [7]. There have also been similar initiatives in [8] and [9].

Interestingly, we observed that all previous work on combining security and context-aware computing follow the same pattern: using contextual information to enrich the access control model in order to secure context-aware applications with a focus on specific applications.

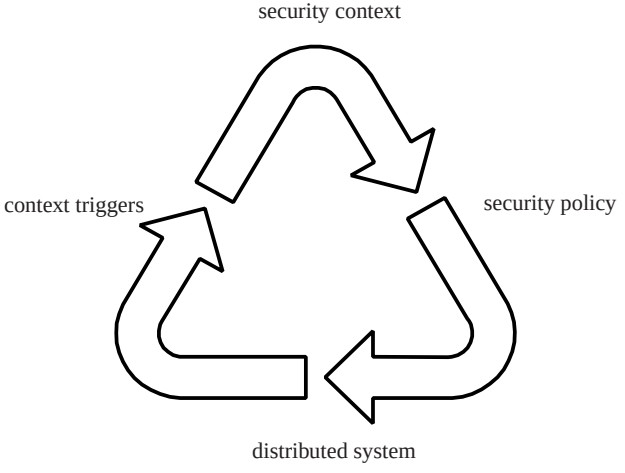

Fig. 1. Context-based security

3.2 Contributions of This Paper

By comparison with previous contributions discussed in Section 3.1, our work is about contextualizing security rather than securing context-aware applications. Even if the difference is not completely apparent actually where we begin by describing the overall architecture, fundamental differences will emerge in future

contributions when detailing each component. As a preliminary example, we cite the new concept of "security context" introduced in Section 4.3.
We summarize in the following the main contributions of our research:

1. To lay down the minimal foundations for a generic context-based security framework with a focus on the software architecture. Generic means to provide the minimal software architecture that can be easily extended to build more specific applications. In addition, context-related modules and resources are loosely coupled allowing the adding/removing of new resources and to modify their respective access policies in a transparent manner. This is an appropriate choice for highly dynamic environments.
2. The second main contribution is to provide a way for managing federations of resources following a specific global policy (an organization's policy for instance) where resources and services join and leave the federation in an ad hoc manner. That is why access policies are organized by resource type and their corresponding actions (see the example in Section 4.1).
3. The third main contribution is to require specific authentication methods depending on a partial context built from the state of the federation.

The resulting prototype will then be designed with the following requirements in mind:

- Provides a framework for the rapid prototyping of context-based security systems,
- Handles both simple and high-level contextual information related to security,
- Easily extensible to manage new protected resources,
- Easily reconfigurable to adapt to new access policies,
- Allows a customizable (context-based) method of authentication (username/password, certificates, etc) by requiring specific credentials depending on a partial context.
- Transparent to both resources and requesting clients; no need to an a priori knowledge of the federation policy.

The following section describes the overall architecture.

4 Context-Based Authorizations Tuning

The term context-aware computing was first introduced by Schilit et al in 1994 [10] as a software that " *adapts according to its location of use, the collection of nearby people and objects, as well as the changes to those objects over time*".

Another given by Dey in [11] states that "*A system is context-aware if it uses context to provide relevant information and/or services to the user, relevancy depends on the user's task.*"

Now context awareness is a well established community with conferences as ubiquitous computing, pervasive computing, etc.

In the field of context-aware computing, context is generally defined as physical parameters (location, temperature, time, etc.) obtained from sensors. However, the user is not really considered in these approaches. In this sense, context is generally managed as a layer between an application and the external world, a type of middleware. Conversely, there is also another approach in which the user (through his knowledge and reasoning) is central in the modelling of context. In this second area, knowledge and reasoning in the accomplishment of a task are described in a context-based formalism, i.e. inside the application itself (e.g. see the contextual graphs in [12]).

Our research covers both the first and second work with an application to the security infrastructure of ubiquitous applications. We concentrate on the main part of the security framework of a distributed system. Namely, access control to shared resources. The framework makes use of RBAC (Role-Based Access Control). Users are affected to roles based on their credentials and competencies [13]. Role-based access is more suitable for pervasive environments since it simplifies the administration of permissions; updating roles is easier then updating permissions for every user individually [14] [15].

4.1 A Case Study

To illustrate the main functionalities of the proposed architecture, we consider a simple example. This example will be developed along with the definition of each component.

We consider a protected document that offers the following operations: *read*, *write* and *delete*. Depending on their credentials and identity, requesting users are a priori affected to one of these two roles: *administrator* or *guest*. The document is available on the network and its access policy is defined and stored in the context engine. In our model, resources are managed by a specific access policy depending on their type; the type of service they provide.

The actual access policy is defined using a rule-based formalism with a simplified grammar (no explicit If/Then clauses). Rule-based reasoning is an area of artificial intelligence (AI) wherein people simulate human behaviors when presented with a new case requiring some action. This approach is used here to specify context-based access policies in order to grant or deny access to resources (see [16] and [17] for more information).

We consider that all protected resources are protected by default, thus, their corresponding policies express only cases when the access is granted (which justified the lack of If/Then clauses). This design choice aims at lightening the process of policies specification. Here is an example:

A Simple Access Policy

```
Resource_type = document;
Action = read ((Role = administrator) OR
         (Role = guest; (Date = Weekdays AND Time = between 8:00-18:00)));
         Authentication = username/password;
Action = write (Role = administrator);
```

```
          Authentication = username/password;
Action = delete (Role = administrator; Date = Weekends);
          Authentication = certificates;
```

Each shared resource defines access rules for each individual operation. The *authentication* tag is used to specify the authentication method required in the actual partial context. The access is granted only when the complete context is build; if the conditions are satisfied and the corresponding authentication phase succeeded. The pattern used above eases further updates of the access policy by adding or removing conditions on it. Defining access policies manually is a cumbersome task in complex real applications with complex relationships among roles. This process can, however, be performed visually using a graphical interface. In [4], Covington et al. propose a graphical policy editor for specifying available roles, their relationships, and policy rules.

Based on the above example, we present the main parts of the security architecture.

4.2 Protected Resource

We consider three types of resources: hardware devices, physical resources (documents, databases) and software resources (operations on a software object or data structure). In order to fit within our model, each resource must respect the following structure (see Fig. 2):

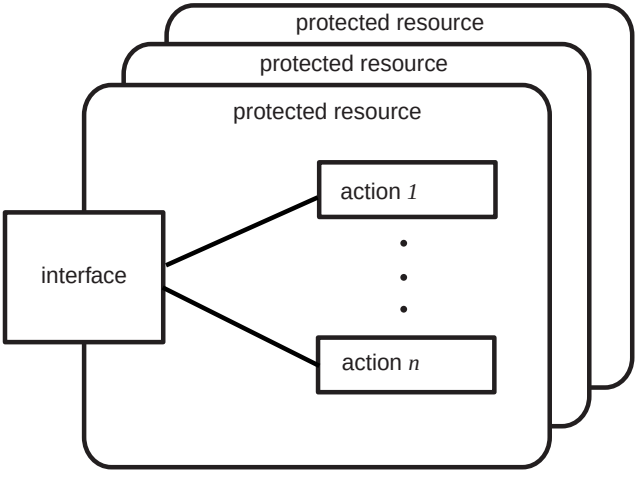

Fig. 2. Structure of protected resources

– Any interaction with the resource is performed via an interface that presents the set of all actions available for the resource,

- Resource actions contain an additional flag that accept only two states: true or false. This flag is used to allow or deny the operation on the resource,
- Each resource is protected by default (all of its corresponding actions have a flag value equal to false).

A user whose role is guest invokes the read action on the document via the interface attached to it. The protected resource identifies the request as coming from a client. It then, forwards it to the context engine.

4.3 The Security Context Engine

The context engine has two responsibilities: modelling contextual entries in order to build a security context and mapping between the security context and the corresponding authorizations on resources. Modelling context requires picking out the most relevant features to reduce it to a meaningful representation [18],[19]. We provide herein two types of classifications of contextual entries depending on their representation aspect and temporal aspect:

Representation Aspect

- ***Simple:*** The collected information is used in its original format. For example, it can represent the value of a parameter,
- ***Interpreted:*** The collected data cannot be used as it is but needs to be converted in a more meaningful format. For example, the contextual entry is "Sunday" that needs to be converted into "Weekday" or "Weekend",
- ***Composite:*** It is a set of simple and/or interpreted entries collected as a whole.

Temporal Aspect

- ***Static:*** It describes contextual information that is invariant, such as a person's date of birth,
- ***Transient:*** The value of a transient contextual entry is updated at run-time and does not need to store information about its past state. For example, time, date, etc,
- ***Persistent:*** Some entries must store historical data about their past state. Persistent contextual entries need to be marked with a time stamp.

Building a Security Context. Our model relies on a set of contextual information relevant to security. This set forms what we call a security context. Designing context in general is not easy and designing a security context suffers the same problem. We present herein an attempt definition of the security context.

> *A security context is a set of information collected from the user's environment and the application environment and that is relevant to the security infrastructure of both the user and the application.*

The word relevant means that has direct or indirect effect on the security policy. In the present work, we are dealing with authorizations, so the contextual information is more precisely relevant to authorizations on available resources. What is really relevant is not fully predictable in advance and depends on the application. This information may include, user identity, membership, resource location, date, time, the user's interaction history with the system, social situation, etc.

In our example, the context engine extracts the operation type and the client's role from the received message. It then builds a partial context based on the client role and the access policy of the requested resource. To build a partial context, the engine retrieves the suggested contextual information from the context bucket. Following the policy defined in section 4.1, requests to the bucket will ask for date and time. The resulting partial context requires a specific authentication method (a username/password method in our example). Thus, credentials provided by the user (username/password, certificates, etc) are additional sources of contextual information. Once the complete security context is built, final actions are performed; access to the document is granted (if authentication succeeded) or denied (if authentication failed). This process is practically equivalent to setting the read operation flag to true or false. Contextual data are received from the security bucket in a primitive type, and then interpreted at the context engine level. For example, date is represented as "*Monday*" and it is the responsibility of the context engine to interpret it as "*Weekday*". This design requirement eases the reuse of the context bucket by different applications with different interpretations of the same contextual data.

4.4 The Security Context Bucket

One of the main problems of context is how to store it and in a way that many applications can use it. This is true especially in distributed applications where both the contextual information and the applications that need it are naturally spread and shared [20]. In order to store contextual entries, we investigate a central point of fall. All the security contextual data are collected into a logical bucket; the security context bucket.

> *The security context bucket is a shared software data structure that offers the notion of container in order to handle the security contextual information.*

A similar approach has been proposed in [21].

At first sight, this approach may seem not very suitable for distributed systems since components interested in context (subscribers) are distributed over different computing devices and developed by different programmers. This incompatibility leads to different interpretations of the same context data. For example, a user's location may be interpreted by one component as a relative distance (near, far) and by another component as an absolute location (using the coordinates). We argue that even if the storage medium is centralized, the interpretation of the selected entries is performed in a distributed manner, at

the component level. In addition, the security bucket acts as a service that has the ability to retrieve a given contextual entry when requested. Thus, multiple security buckets with the same functionality may exist in the network in order to ensure availability and load balancing.

The main advantage of this approach is to ease the scale up of the system for a large number of contextual entries across a wide network. The second advantage is the robustness to failures by making the contextual data available from different places, and finally, to ease the protection of contextual data.

The security context bucket offers the same advantages as encapsulation; a key feature in the object-oriented paradigm. Object encapsulation is also known as data-hiding. It is a software mechanism that protects code and data from being accessed by everyone but only to the methods that need it. In the same manner, the context bucket hides its content, and access to it (read and write operations) is subject to a security policy that manages interactions with clients.

Context entries are collected from the distributed network by the mean of a group of agents. Gathering agents are mobile [22] and launched by the security bucket when requested. Their main role is to collect needed contextual data from their remote location, by requesting sensors, software applications and environment. The content of the context bucket is primordial in configuring the security policy of the system. This content must then be protected.

Protection and Privacy of the Security Context Bucket. Designing context-based security systems poses a kind of tricky issue. The more a context-based security system knows about the user's and the application environment, the more it can provide fine-grained access control to protected resources. On the other hand, it becomes easier for hackers (at least theoretically) to compromise the security of the system not directly (by attacking resources) but may do it indirectly by providing false contextual data to the bucket or by accessing critical users information contained in it. The first can be achieved by launching malicious gathering agents that provide corrupt data and the second can be achieved by accessing critical information from the context bucket, such as users' private data.

Thus, and in order to achieve protection and privacy of the security context, an additional component is then required in our architecture. The authentication module authenticates both entities that provide contextual entries (gathering agents) and entities that need access to the security context (context engines).

However, in case of the unavailability of a contextual entry, the system must be able to learn from previous experiences and propose an alternative.

Collected contextual information are used by the context engine (described in section 4.3) in order to build a security context and then to deduce the actions to perform. The security bucket requirements are summarized in the following:

- The security context bucket has the ability to create, manage and authenticate gathering agents,
- Contextual entries are sensed and stored in a primitive format that eases all possible interpretations,

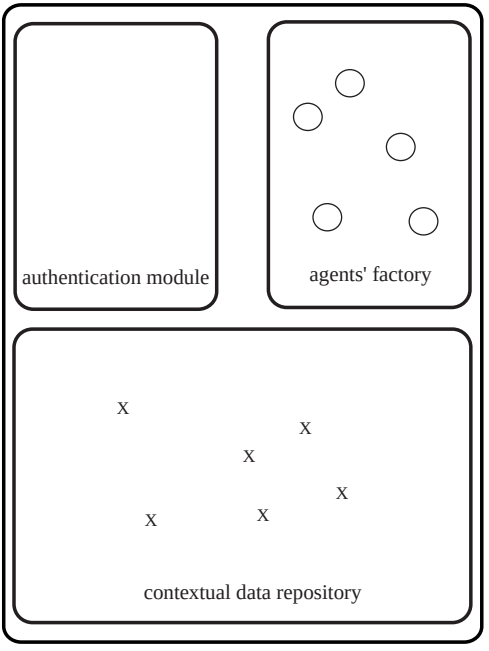

Fig. 3. The security context bucket

- Only authorized sources (gathering agents) are able to add/update data in the security context bucket,
- Only authorized destinations (security context engines) are able to read the bucket content,
- The security context bucket must maintain historical information at the finest level of detail possible of its content.

Figure 3 illustrates the main parts of the security context bucket:

1. The agents' factory produces gathering agents in order to collect contextual data upon request,
2. The contextual data repository is used to store gathered contextual data,
3. An authentication module is also needed in order to authenticate gathering agents and requesting context engines.

The following figure (Fig. 4) illustrates the overall structure of the framework and the relationships among the different components. Further changes in the access policy of protected resources can be transparently performed by updating the corresponding policy in the context engine. Resource can join or leave the distributed infrastructure without disturbing the security infrastructure.

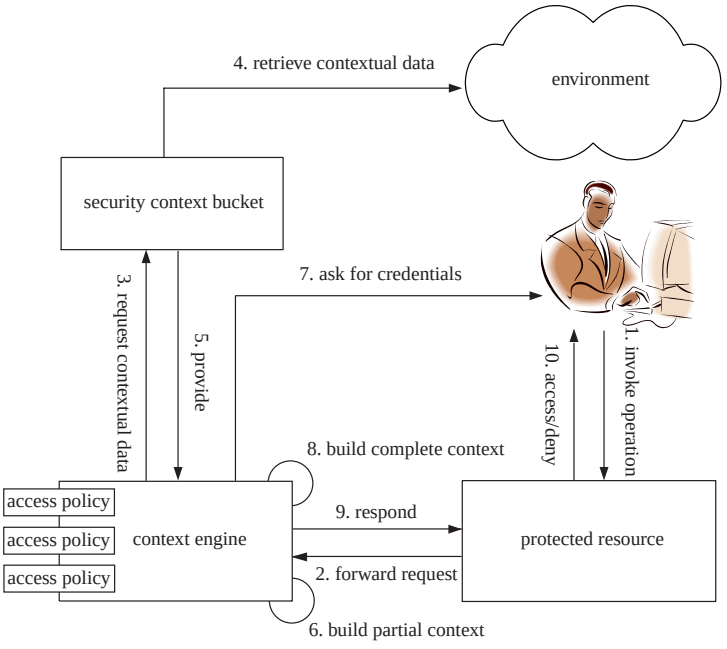

Fig. 4. Overall architecture

5 Conclusion and Future Work

Security systems for distributed infrastructure are generally bound to static access policies that make them very difficult to adapt to new threats. This situation is due to the lack of consideration for context in existing security systems. Context-based security is a recent research direction that aims at providing flexible security models for distributed infrastructures, where the user's and application environments are continually changing.

In this paper we have introduced a new model for context-based authorizations tuning in distributed systems. Much of this work is focused on providing a generic minimal architecture based on loosely-coupled components. The architecture provides tools for collecting and modelling security contextual data. We have introduced the concept of partial context and illustrate how it can be used to request specific authentication methods in order to control access to protected resources. In the near future, we intend to extend the proposed framework to handle inaccurate or unavailable contextual data, specify a registration protocol that allows adding context-based access policies in the context engine and investigate the case of complex relationships between user's roles. The use of contextual graphs [23] [24] is also a potential methodology for modelling the security context; in order to access a resource, the specification of an exhaustive graph may prevent frauds by including only "*safe*" cases.
We are actually investigating a test-bed application with a federation of services

inside a university network. Both tutors and students provide plug and play services to be used in a safe environment. These services (or resources) include: course subscription services, online exercises, chat systems, printing service, etc. Our model provides an administrative tool to manage authorizations on these resources based on the user's role (regular student, auditor, tutor or guest) and the context of interaction (time, day, history of the user's use of the service, etc). We believe that this approach will prove to be an interesting starting point for further investigations of flexible security models for next-generation distributed authorizations frameworks.

References

1. Brézillon, P., Pomerol, J-Ch.: Contextual Knowledge Sharing and Cooperation in Intelligent Assistant Systems. Le Travail Humain 62 (3), PUF, Paris, (1999) 223–246
2. Balfanz, D., Dean, D., Spreitzer, M.: A Security Infrastructure for Distributed Java Applications. In Proceedings of the 2000 IEEE Symposium on Security and Privacy, Oakland, California, (2000) 15–26
3. Wack, J.: Guidelines on Firewalls and Firewall Policy. Computer Security Division. Information Technology Laboratory. National Institute of Standards and Technology
4. Covington, M.J., Ahamad, M., Srinivasan, S.: A Security Architecture for Context-Aware Applications. Technical Report GIT-CC-01-12, College of Computing, Georgia Institute of Technology, May 2001
5. Covington, M.J., Fogla, P., Zhan, Z., Ahamad, M.: A Context-Aware Security Architecture for Emerging Applications. In Proceedings of the Annual Computer Security Applications Conference (ACSAC), Las Vegas, Nevada, USA, December 2002
6. Covington, M.J., Long, W., Srinivasan, S., Dey, D., Ahamad, M., Abowd., A.: Securing Context-Aware Applications Using Environment Roles. In Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT '01), Chantilly, Virginia, USA, May 2001
7. Masone, C.: Role Definition Language (RDL): A Language to Describe Context-Aware Roles. Dartmouth College, Computer Science. Hanover, NH. TR2002-426. May 2002
8. Shankar, N., Balfanz, D.: Enabling Secure Ad-hoc Communication Using Context-Aware Security Services. Extended Abstract. In Proceedings of UBICOMP2002 - Workshop on Security in Ubiquitous Computing
9. Osbakk, P., Ryan, N.: Context Privacy, CC/PP, and P3P. In Proceedings of UBICOMP2002 - Workshop on Security in Ubiquitous Computing
10. Schilit, B.N and Theimer, M.M.: Disseminating Active Map Information to Mobile Hosts. IEEE Network, 8(5): (1994) 22–32
11. Dey, A.K.: Understanding and Using Context. Personal Ubi Comp 5 (2001) 1, 4–7
12. Brézillon, P.: Using Context for Supporting Users Efficiently. Proceedings of the 36th Hawaii International Conference on Systems Sciences, HICSS-36, Track "Emerging Technologies", R.H. Sprague (Ed.), Los Alamitos: IEEE, (2003)

13. Georgiadis C., Mavridis I., Pangalos G., Thomas, R.: Flexible Team-based Access Control Using Contexts. In Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT 2001) ACM SIGSAC, Chantilly, VA, U.S.A, May 2001
14. Ferraiolo, D.F. and Kuhn, D.R.: Role Based Access Control. In Proceedings of the 15th National Computer Security Conference (1992)
15. Sandhu R. S., Coyne E. J., Feinstein H. L., and Youman C. E.: Role-Based Access Control Models. IEEE Computer, Volume 29, Number 2, February 1996, 38–47
16. Clancey, W. J.: The Epistimology of a Rule-Based Expert System: A Framework for Explanation. Artificial Intelligence Journal, 20(3): (1983) 197–204
17. Clancey, W. J.: Notes on "Epistimology of a Rule-Based Expert system". Artificial Intelligence Journal, 59: (1993) 197–204
18. Kokinov, B.: A Dynamic Approach to Context Modeling. Proceedings of the IJCAI-95 Workshop on Modeling Context in Knowledge Representation and Reasoning. LAFORIA 95/11, (1995)
19. Bouquet, P., Serafini, L., Brezillon, P., Benerecetti, M., Castellani, F. (eds.): Modelling and Using Context. In Proceedings of the 2nd International and Interdisciplinary Conference, CONTEXT99, Lecture Notes in Artificial Intelligence, Vol. 1688, Springer Verlag, (1999)
20. Henricksen, K., Indulska, J., Rakotonirainy, A.: Modeling Context Information in Pervasive Computing Systems. In Proceedings of the 1st International Conference, Pervasive 2002 - Zurich August 2002, Lecture Notes in Computer Science, Vol. 2414, Springer Verlag, (2002) 167–180
21. Hong, J.I.: The Context Fabric: An Infrastructure for Context-Aware Computing. Extended Abstract. Conference on Human Factors and Computing Systems, (2002) 554–555
22. White, J.E.: Mobile Agents. In I. Bradshaw and M. Jeffrey, editors, Software Agents. MIT Press and American Association for Artificial Intelligence, (1997)
23. Brézillon, P., Pasquier, L., Pomerol J-Ch.: Reasoning with contextual graphs. European Journal of Operational Research, 136(2): (2002) 290–298
24. Brézillon, P., Cavalcanti, M., Naveiro, R., Pomerol J-Ch.: SART: An intelligent assistant for subway control. Pesquisa Operacional, Brazilian Operations Research Society, 20(2): (2002) 247–268