Context-Based Constraints in Security: Motivations and First Approach

Ghita Kouadri Mostéfaoui 1

Department of Informatics
University of Fribourg
Fribourg, Switzerland

Patrick Brézillon 2

LIP6
Université Paris 6
Paris, France

Abstract

Recent research in security shows a real need for contextual information in building highly adaptive security systems. The need for such security for web services is fast becoming imperative. In this paper, we review the main contributions in this domain. Based on this study we present a glimpse at the CoDiS project; a generic layer for the rapid prototyping of context-based security systems, aiming at getting around the limitations of existing systems.

Keywords: Context, security, distributed applications, design patterns, software frameworks.

1 Introduction

Security is now one of the main challenges that cross disciplines, it is applied at different levels: network level, application level, user level, etc. Security is also a never-ending process that requires dynamic reconfiguration of its components in order to cope with the constantly changing requirements of highly dynamic environments. Also, traditionally, security systems tend to
prevent unauthorized access but there is no effective way to deal with users’ ”bad” intentions when they have the right credentials.

The problems discussed previously suggest the need for constraining actions with their respective context of occurrence. For example, an e-mail may be relevant to user a while it is considered as a junk email for user b. Taking context - here, the user’s profile - into consideration by the spam filter is the only way to resolve the conflict. More specifically, Web services are subject to personalization because of the need of accommodating user preferences. Securing such applications is itself subject to adaptive behavior according to the security context.

In this paper, we review the main contributions on combining context and security and discuss why this combination is of interest for web services. Based, on this study, we draw a set of conclusions for which aim is to identify limitations of previous contributions and to justify the need for a generic framework for managing context-based security. A summary of the CoDiS project (a generic framework for managing context-based security in distributed systems) is presented. Section 2 details some relevant scenarios in order to show why context is important for better security for web services. Section 3 discusses previous contributions on using context in security systems - as a general paradigm - and details limitations of previous contributions. The CoDiS project is introduced in Section 4. Some frequently questions about CoDiS are answered in Section 5. Section 6 concludes this paper.

2 Why Context is Important for Better Security for Web services

Where traditional security is mainly based on the concept of roles and tasks affected to these roles, we argue that the integration of context into these security solutions is a new step towards future more effective security. We discuss in the following the need for making security for web services a context-based activity. Thus, and in order to illustrate the relevance of our approach, we present two scenarios that justify the need for context in security for web services.

2.1 Enforcing web services semantics or fine-grained access control

Current applications are taking benefit of the pervasive computing paradigm which pushes computations from a computer into everywhere but in the same time quickly increases complexity, openness, interconnection, and interdependence [1]. These properties have made these systems more vulnerable and
difficult to protect than ever and new types of security attacks based on malicious behavior emerged. As an example of such attacks are those carried out by intruders who possess legitimate access to the system and act within their bounds of normal behavior, but who are actually abusing the system. For example, consider a web services-based e-business application which offers access to a database containing business-related data. The major characteristic of such a web service is to accept different kinds of users (administrator, local employees, remote clients, etc) logging from various locations. An operation like update the database is legitimate for each type of users but performed by a remote client on a weekend is suspicious and may harm the system. In these cases, the security of the web service must conform to its specific semantics. These semantics are dependant on a set of contextual information such as time, number of performed operations, interaction history, etc. More complex scenarios may be defined and more high level input may influence these scenarios. Contextual information must then be taken into account in defining the fine-grained access policy.

2.2 Determining security levels in heterogeneous applications

Heterogeneous networks are widely available and they provide a platform for web services interactions to take place. Theses networks have many different properties such as transmission speed, communication media (RF, Infrared, Microwave, etc), connectivity, bandwidth, range, etc. Moreover, many types of computing devices are widely used and they have diverse capabilities. Additionally, exchanged information between web services and their clients (users, other web services) range from simple text messages to video streams. Each type of information requires the use of a specific security protocol in order to ensure integrity for example. To secure this diverse environment, we should adapt several security levels dynamically according to the diverse networks and computing devices used to interact with web services, where adaptability is based on the context of use. In this specific example, context helps enforcing availability which is one of the main security goals as stated in [2].

3 Context-Based Security as a General Paradigm

3.1 Related Contributions

The literature on combining context and security mainly concentrates on context-based RBAC (Role-Based Access Control). Primarily, these contributions aim at extending the traditional RBAC model by integrating a set of constraints. In this perspective, Masone [6] designed and implemented RDL
(Role-Definition Language), a simple programming language to describe roles in terms of context information. RDL has been designed for simplicity of use and extensibility. A similar work is described in [7]. McDaniel in [8] proposes an extended form of authorization policies. In this extended form, conditions corresponding to context are expressed using programs rather than expressions defined over a fixed set of attributes such is the case in most contemporary systems.

Georgiadis et al. integrate their team-based access control (TMAC) with contextual information [9] which results in a system called C-TMAC. The TMAC model aims at providing an access control model that supports collaborative activity being accomplished by teams of users. C-TMAC is a hybrid access control model that takes advantage of role-based permission assignment of RBAC and yet provides the flexibility for fine-grained activation of permissions for individual users on individual object instances. This fine-grained access control is dictated by a set of constraints (or context). For example, in a healthcare setting a doctor may have the permission to prescribe certain medications. However, the doctor should not be allowed to prescribe for anyone. Rather, he/she should be allowed to prescribe only for the patient’s he/she is taking care of.

An interesting work by Covington et al. in [4] and [5] describes an approach to design security services that incorporate the use of security-relevant ”context” in order to provide flexible access control. The authors target intelligent home environments such as the Aware Home [3]. In order to facilitate the collection of environment variables and their associated values, this approach makes use of the context toolkit [10] which aims at facilitating the development and deployment of context-aware applications. Access policies are based on environment roles [4] and encoded into the eXtensible Markup Language (XML). In regards to combining context with a security service other than access control, the efforts are rarer. It is worth mentioning the work by Persson et al. in [11] which introduces context-agile encryption as a technique to provide a different encryption (in terms of key lengths, type of encryption algorithm, etc) according to the applications needs and to hardware capabilities. For example, some web services in high speed communication networks may be able to tolerate long times to encrypt/decrypt information, but may also need to protect that information for a long period of time. Other types of web services, dealing with data that is sensitive while useful, but quickly becomes stale, might benefit from short encryption/decryption times that may accompany a less cryptographically robust algorithm.

Bellavista et al. propose a context-centric access control middleware for mobile environments called COSMOS [12]. The middleware dynamically de-
terminates the contexts of mobile proxies, and rules the access to them based on a set of data such as user profiles and system/user-level authorization policies. COSMOS has been tested in the design and implementation of a context-centric movie assistant which aim is to allow mobile users to find nearby cinemas and to exchange opinions about cinema characteristics such as seat comfort, air conditioning, sound and screen resolution. Other contributions can be found in [13], [14] and [15].

3.2 Limitations of Previous Contributions

Even if context has been used for some time in policies specification, it is rarely considered explicitly. As a consequence very few works benefit from the theories and tools already developed in the context-aware computing area in order to model the needed contextual information. The other observation is that most policies rely on location and date as contextual information and do not include high level abstractions such as the network state and the user’s interaction scheme with the system, and which could be of a great importance in managing compositions of web services. When it is the case, it is not always clear how people deal with these aspects. Additionally, even if the rule-based representation is often adopted as an intuitive solution for modeling context, such as in [4] and [6], it suffers from three main limitations. The first one is the difficulty to maintain such formalisms in case of complex systems to secure. The second limitation is the difficulty to identify all the needed contextual information from the rule-based formalism which makes the context management task awkward. The third main limitation is that it does not provide a convenient way for understanding the followed strategy of the policy and makes the security management task cumbersome for security administrators. Decision trees are another way to structure the rules. However, the fine-grained nature of context leads to a combinatorial explosion of the trees size. Thus, a more convenient way for modeling context should be applied.

Despite the advantages and limitations of previous systems, the context existing systems relies on is static, which means that context-based security actions are supposed to be known a priori as is the case in ancient expert systems. The resulting security systems are thus, not able to learn from previous failures.

Finally, and since previous systems rely on specific applications, there is no generic software solution that enjoys reusability and extensibility features and that may serve as a platform for building different domain-specific context-based security systems such as the ones for web services.
4 A Glimpse at CoDiS: A Generic Layer for the Integration of Context-Based Security in Distributed Systems

In order to respect the workshop’s number of pages limit, note that this section is intended to introduce the CoDiS project in a concise manner. Thus, we don’t emphasize on technical details that may be found in [17].

Based on what we have seen in the previous Section, our motivations to develop the CoDiS project are to satisfy this main set of objectives:

(i) to lay down the minimal foundations of a generic framework for the rapid prototyping of context-based security systems with a focus on the software architecture. Generic means to provide the core architecture that can be easily extended or customized to build more specific applications such as web services,

(ii) to decouple context acquisition and management from context modeling and context usage,

(iii) to provide an easy way for switching between different context-based policy modeling techniques,

(iv) to support the dynamic aspect of contextual information,

(v) support easy modifications of context-based security policies.

CoDiS stands for context-based security in distributed systems. The aim of this project is not to target a specific security problem such as context-based access control, but rather to provide a generic implementation of a context-based security system that may be used in any security infrastructure highly dependent on contextual information. CoDiS is the realization and evolution of the first sketch of the project presented in the Context’03 conference (see [16]).

We focus more on the implementation of a security context including its acquisition, storage, representation, modeling and integration within a pervasive environment.

The current implementation of CoDiS is a pattern-oriented framework for context acquisition, representation and modeling specifically dedicated to security systems. Figure 1 illustrates the overall architecture of the framework. Two main modules are shown:

(i) The context bucket is responsible for context lookup, storage and provision.

(ii) The context engine is responsible for context representation (aggregation, composition and interpretation) and for context modeling. The last
task is performed in order to map each security context with its corresponding security actions. Figure 1 depicts the relationships between the two modules, and shows the different interactions between them in order to provide context-based security to distributed resources served by web services.

4.1 CoDiS design principles

CoDiS has been designed following a set of guidelines with an aim to support a generic layer for the rapid prototyping of context-based security systems. The discussion of the in-depth software architecture of CoDiS is out of scope of this paper and will not be discussed here; however, we refer the interested reader to [17] for a more detailed discussion.

Our concern for CoDiS software quality, however, suggests satisfying a set of design principles. The following sections detail each one of these principles.

4.1.1 Object-oriented framework design

The CoDiS architecture is composed of distinct parts each one packaging either the core parts, or the customizable parts of the architecture that are frequently extended in order to fit within applications needs.

This requirement aims at reducing software complexity and is satisfied using (a) a layered architecture and (b) a pattern-oriented design. Figure 2 shows a sample example; a simplified class diagram of the context representation module where core parts of the architecture emerge from the customizable parts (specific contextual entries, specific interpreters of contextual entries). The upper layer defining the core part of the architecture is designed following a well known design pattern; the composite (see [17] page 163).

4.1.2 Technology independence

This design principle aims at isolating the code related to a specific technology from the core packages of the framework. The distributed nature of our architecture requires the use of some protocols and mechanisms in order to exchange messages and information. Instead of reinventing the wheel, we investigated many technologies offering these capabilities. The current version of CoDiS makes use of the discovery and communication protocols of the Jini technology [19] in order to support context discovery and gathering. Jini capabilities are assembled in a specific package and its corresponding mechanisms are separated from the core architecture as much as possible in order to make
4.1.3 Supporting the dynamicity of security contexts

On the other hand, since the framework is intended to be generic and customizable for different scenarios, the main particularities of CoDiS is the support for the dynamic aspect of security contexts. The need for the support of context dynamicity is justified by the following; a security context can be built during a lap of time (for example in the case of access control) or it can be built only after a longer period (for example when a security context corresponds to the detection of abnormal behaviors in a secure building, where this behavior is the accumulation of a set of specific actions performed during a longer period of time).

After reviewing the main context modeling techniques [20], we retained
contextual graphs [21] as a modeling approach for context-based security policies. In both cases - discussed earlier - contextual graphs policies provide an explicit support for context dynamicity which allows period-independent construction of security contexts. Thus, offering a convenient way for observing the evolution of security contexts.

Contextual graphs-based policies contain exactly one input and one output, and a general structure of spindle. A path from the input to the output of the graph represents a practice. Contextual graphs provide a context-based representation of a task execution. More clearly, they express the set of multiple alternatives (practices) to execute a task. An alternative is executed among the others according to the current context.

Figure 3 illustrates an example contextual graphs policy that manages fine-grained access control to healthcare records in a hospital served by services. Records are stored in a database. The hospital hosts a distributed environment that allows accessing the records from remote terminals by invoking the

---

3 CoDiS is designed in a way that it seamlessly accepts the inclusion of any new modeling approach of security policies. Contextual graphs have been retained for the reasons mentioned in the paper.
appropriate web service. For sake of clarity, only a small part of the contextual graph is represented in the figure.

Hospital staff wishing to access a patient’s health care record first invokes the web service by authenticating herself as being a member of the staff. Then, depending on the role of the user (C1), additional contextual information infers the decision of authorizing access to the record. For example, if the user is the treating physician (C2.1), an additional password authentication is required when the physician is connected from a terminal inside the hospital (C4.1). This additional step is not required if the physician requests the record from the emergency room (C4.0). In the case the physician requests access to the record from a terminal outside the hospital, an additional operation is performed in order to encrypt the communication between the two parties. This is done due to the high sensitivity of the information contained in health care records.

As one can observe, for a nurse, only two cases are possible in contrast to a doctor (treating physician or not). A nurse can access the patient’s record if the request to the web service is performed from an emergency room or
if it is performed from a terminal inside the hospital. Requests made from outside the hospital are not authorized. Unauthorized path are not shown on the contextual graph. This is to allow specifying only safe paths in order to perform a secure action and is commonly known as closed security policy “which is not explicitly permitted is denied”.

Contextual graphs provide a clear representation of security policies and of the constraints that condition their behavior. Security mechanisms and security contextual information are expressed using a set of graph elements see Figure 4. At a contextual element - represented by a circle - a diagnosis is made in order to evolve a security context. For each security context a set of security mechanisms are enforced. Security actions are represented by rectangles. Recombination nodes, represented by black circles play the role of a tie. More clearly, when reaching a recombination node, the security context is degenerated which means that the needed actions have been already applied and the context is no longer relevant. Arrowed lines represent parallel action grouping; when two or more actions can be realized in parallel. Macro-actions represented in the contextual graph by MAi are special cases of activities reduced to a simple sequence of actions.

Contextual graphs distinguish between three types of context namely, contextual knowledge, proceduralized context and external knowledge. In order to illustrate these three types of context, we consider a decision making process, whose actions to activate depend on a set of contextual information. The process is modeled using contextual graphs. Thus, contextual knowledge is a subset of context that directly intervenes in the decision making process. Moreover, only a subset of the contextual knowledge is used at a given step in the decision making process, this is known as proceduralized context. The remaining subset which includes information which is not relevant to the situation is called external knowledge.

Again, consider the example of Figure 4. The context of the action A3 in the previous example is described in a fixed and static way. Once the action A3 is executed, the value C3.1 of C3 does not matter anymore. The contextual element C3 leaves the proceduralized context at the recombination node R3 to go back to contextual knowledge. Thus, the context of the action A8, which follows the execution of the action A3, is described by:

- the proceduralized context: C1 with the value C1.1, and C2 with the value C2.0, and
- the contextual knowledge: C3, C4, C5, C6, C7.

The context of the action A8 is also described in a fixed and static way. It differs from the context of action A3 by the contextual element C3 that moved
Fig. 4. An example contextual graph

from the proceduralized context of the practice to the contextual knowledge. The contextual knowledge and the proceduralized contexts of actions A3 and A8 are different, but the sum of the contextual knowledge and the proceduralized context is constant at the level of the contextual graph. The dynamics of the context appears at the practice level. The contextual knowledge and the proceduralized context evolve during the application of a practice (along a path). For example, consider the upper practice in Figure 4: A0, A4, A5, A9, A10. Its context presents the following dynamic along the practice application (each line of the Table 1 represents a step in the application of the practice, a step corresponding to a change in the context): The movement inside the context (and its dynamic from an outside viewpoint) arises from a contextual element entering the proceduralized context by its instantiation at a contextual node, or, conversely, the withdrawal of the instantiation of a contextual element (becoming again a piece of contextual knowledge) at a recombination node. The movement between the proceduralized context and the contextual knowledge follows the rule “last in, first out.” Thus, two contexts having the same contextual knowledge and proceduralized context (as at lines 2 and 4 of the Table above) are different by their history in the practice.
4.1.4 Supporting incremental acquisition of practices

By their nature, contextual graphs-based policies support incremental acquisition of practices. Contextual graphs have the capacity of evolving by accommodation and assimilation of practices. Generally, the new practice differs by few changes (an action instead of an existing one, the addition or lack of an action). Then, the CxG system enters a phase of acquisition of the new practice from the security administrator. The practice acquisition concerns the new action to integrate and the contextual element that discriminates that action with the previous one. The integration of the new practice requires either the addition of a new branch on an existing contextual node (just before the diverging part of the practice), or the introduction of a new contextual node to distinguish the alternatives. In all the alternatives, the piece of contextual knowledge to add must be instantiated.

5 Q & A about CoDisS

Due to the number of pages limit, some details have been omitted. This Section is intended to answer the most frequently asked questions about CoDiS in a short manner.

(i) How to obtain context to consider in contextual nodes?
Contextual nodes are diagnosis nodes where a specific alternative is selected among a set of other alternatives according to the value of the context. According to the application domain, context may be obtained locally such as time of request or from a client such as his preferences or through sensors such as the temperature of a room. The last type of
context is obtained using an agent-based approach for context discovery, collection and storage (see Figure 1, Context-bucket layer). Collected information from different sources may also be concatenated to form a composite context.

(ii) What is the main added value of the framework compared to previous work?

Compared to other works such as the Context Toolkit [10], and from a software engineering perspective, CoDiS relies on design patterns and on a software framework approach for building highly reusable and maintainable software architectures. Rather than being tied to a specific application, CoDiS may be easily customized to manage any type of context and can be used for the rapid prototyping of context-based security systems.

(iii) What in CoDiS is specific to security? Why isn’t CoDiS broader than that?

Initially, CoDiS has been designed for security matters where context should be taken into account. The choice of the policies modeling technique has been inspired by the need for handling the dynamic nature of security and all the interactions of the CoDiS components are secured. However, the generic approach for building the software architecture makes it a good candidate for the rapid prototyping of context-based systems in the broader sense.

6 Conclusion

"A law is made to be circumvented" this is what we learn in schools of laws. Computer security hackers are not different from criminals and operate by getting around safe security contexts. Context-based security as we present it is a radical approach to security and aims at systematically integrating context in different security services such as cryptosystems, access control and adaptive authentication.

Web services are now subject to many research on their discovery, composition, self-coordination, personalization and security. Security for web services is however, generally addressed through a set of security standards which are sometimes competitive and overlapping. Currently, there are no definitive guidelines for applying these standards, especially when context should be taken into account.

We discussed in this paper context-based security as a general paradigm, highlighted its relevance to security for web services and discussed the limitations of previous work. The CoDiS project was motivated by the need for a generic framework for the integration of contextual information into security
solutions. It also aims at getting around limitations of previous contributions by essentially (a) considering the dynamic aspect of security contextual information, and (b) supporting incremental acquisition of practices. However, we attract the reader attention to the main benefit of CoDiS which is the support for a global view of context-based security; rather than being limited to access control, it provides a generic platform upon which customizable adaptive security services can be built for emerging applications such as web services.

Many open issues still exist in adaptive security, the main ones are: dealing with missing or unambiguous context, resolving conflicts in security policies and automatic learning of context-based security policies.

References


