SECURITY AND DEPENDABILITY IN AMBIENT INTELLIGENCE SCENARIOS
The Communication Prototype

Alvaro Armenteros
Security Products for Bussines, Telefónica I+D, Madrid, Spain

Antonio Muñoz, Antonio Maña, Daniel Serrano
Department of Computer Science, Universidad de Malaga, Spain

Keywords: Security patterns, Security services, Ambient Intelligence.

Abstract: Ambient Intelligence (AmI) refers to an environment that is sensitive, responsive, interconnected, contextualized, transparent, intelligent, and acting on behalf of humans. Security, privacy, and trust challenges are amplified with AmI computing model and need to be handled. Along this paper the potential of SERENITY in Ambient Intelligence (AmI) Ecosystems is described. Main objective of SERENITY consists on providing a framework for the automated treatment of security and dependability issues in AmI scenarios. Besides, a proof of concept is provided. In this paper, we describe the implementation of a prototype based on the application of the SERENITY model (including processes, artefacts and tools) to an industrial AmI scenario. A complete description of this prototype, along with all S&D artefacts used is provided in following sections.

1 INTRODUCTION

One of the key aspects of the new emerging environments of Ambient Intelligence is Security. Dependability for these environments is also an important feature to be considered. In this paper, we will consider the problems and challenges arising concerned to security and dependability in Ambient Intelligence. We will provide an overview of the current solutions proposed for them as well as an instance of a real world scenario related to AmI where those solutions are applied to cover its S&D requirements as a practical way to validate them and explore challenges arising for these kinds of scenarios.

This paper is structured as follows: section 2 is a background. Section 3 introduces some considerations of security and dependability in AmI scenarios. Section 4 presents the SERENITY (SERENITY project, 2006) approach. In section 5 we describe a proof of concept on a wireless communication scenario. Finally section 6 some conclusions.

2 BACKGROUND

Among the more relevant approaches for modelling security and dependability aspects in Ambient Intelligence (AmI) ecosystems found in literature, we highlight those based on Components, Frameworks, Middleware, Agents and the enhanced concept of Pattern. Concerning components, these capture expertise in the form of reusable software elements to solve a problem under a set of context conditions, provided by a set of well defined interfaces and an associated description of their behaviour (Merabti et. al, 2004; Zhang Shi, 1998).

Middleware based approaches capture expertise in the form of standard interfaces & components. Then applications developers are provided with a simpler aspect to access a set of specialized, powerful and complex capabilities. However, some issues arise such as the high computational cost of the middleware components, especially for those small devices involved in AmI ecosystems with limited capabilities, as well as the security infrastructure of middleware systems, which is restricted to authorization and access control in most cases (BEA White Paper, url; Object Management Group, url). The potential use of Frameworks is found in developing secure services (Wilson et. al, 2003; Sanchez, 2007). In (Sampemane et. al, 2004) a framework that uses ontologies and the Common Criteria classification of security requirements are described. This is intimately related with the main approach of this paper, where the use of ontology facilitates reasoning about the requirements and also reusability of goal and domain knowledge across a large body of software developers. From a different perspective there is the Agent paradigm, which is especially well suited for highly distributed environments such as AmI scenarios thanks to properties like: autonomy, interaction, context awareness and goal-oriented nature. But in case of modelling security aspects are much more limited (Boudaoud et. al, 2002) because an agent is an independent entity and many security solutions can not be represented as agents.

The concept of security pattern was introduced to support the system engineer in selecting appropriate security or dependability solutions. However, most security patterns are expressed in textual form, as informal indications on how to solve some (usually organizational) security problem (IBM's Security Strategy team, 2004; Yoderand et. al, 2000). Some of them make use of more precise representations based on UML diagrams (E. B. Fernandez, 2000). Perhaps the first and the most valuable contribution as pioneer in security is the work from Joseph Yoder and Jeffrey Barcalow (Yoderand et. al, 2000), a natural evolution of this work is presented by Romanosky in (S. Romanosky, 2001). Eduardo B. Fernandez in his work about authorization patterns (E. B. Fernandez, 2000) proposes a further step in the abstraction of patterns. In (Fernandez et. al, 2001) authors propose the decomposition of the system into hierarchical levels of abstraction.

Other authors propose other alternatives to provide formal characterizations of patterns. The idea of precisely specifying a given class using class invariants and pre- and post-conditions (Soundarajan, e. al, 2006). Also Mikkonen in (T. Mikkonen, 1998) focus his approach on behavioural properties. In this approach data classes are used to model role objects, but guarded actions (in an action system) are used to model roles methods.

3 SOME CONSIDERATIONS OF S&D IN AMI SCENARIOS

The Information Society Technology Advisory Group vision is that AmI applications will be influenced by the computational, physical and behavioural contexts that surround the user (for instance, because of resource availability and security or privacy requirements). The concepts of system and application as we know them today will disappear, evolving from static architectures with well-defined pieces of hardware, software, communication links, limits and owners, to architectures that will be sensitive, adaptive, context-aware and responsive to users' needs and habits. AmI ecosystems offer highly distributed dynamic services in environments that will be heterogeneous, large scale and nomadic, where computing nodes will be omnipresent and communications infrastructures will be dynamically assembled.

AmI environments impose some constraints in the connectivity framework, power computing as well as energy budget. This makes of AmI a significantly different case within distributed systems. The combination of heterogeneity, dynamism, sheer number of devices, along with the growing demands placed on software security and dependability (S&D), make application development vastly more complex. Also, the provision of security and dependability for applications becomes increasingly difficult to achieve with the existing security engineering mechanisms and tools.

In the new AmI scenarios, not only systems as a whole but also individual applications running in or supported by those systems will have to adapt to dynamic changes to hardware and software, and even firmware configurations, to unpredicted and unpredictable appearance and disappearance of devices and software components. In other words applications must be able to adapt dynamically to new execution environments. As a consequence pre-defined trust relationships between components, applications and their system environments can no longer be taken for granted. Therefore, the increased complexity and the unbounded nature of AmI applications make it impossible, even for the most experienced and knowledge-able S&D engineers, to foresee all possible situations and interactions which may arise in AmI environments and therefore create suitable solutions to address the users' security and dependability requirements. Additionally S&D engineers will be faced with pieces of software, communication infrastructures and hardware devices not under their control. Thus, approaches based on the application-level security will not be sufficient to provide security and dependability to the AmI eco system as a whole.

An AmI environments relevant feature is that they will contain a large number of heterogeneous computing and communication infrastructures and devices that will provide new functionalities, enhance user productivity, and ease everyday tasks. These devices will hold a variety of data with different security and privacy requirements. This information will be used in different ways in different applications and computing contexts and, therefore, different policies (possibly contradicting) will be applied. Hence, in such settings, securing the device or the information alone or even each individual application is not sufficient, and context information should be integrated in order to be able to choose appropriate security mechanism on-the-fly.

Because of their complexity, and because elements will be under the control of different owners, security mechanisms will need to be supervised (monitored) in order to identify potential threats and attacks and decide on recovery actions, if possible. Thence some existing approaches can provide suitable solutions to sup-port the dynamic evolution of security policies for specific security mechanisms at particular system operation layers (application, networking). However, these approaches cannot be extended to support the dynamic evolution of general security mechanisms (as opposed to security policies for a single mechanism). Furthermore, their results are extremely complicated to integrate, monitor and dynamically evolve as would be required by AmI ecosystems. For the very same reasons, S&D approaches for AmI ecosystems cannot hope to synthesize new S&D mechanisms or new combinations of these mechanisms fully automatically and dynamically. Thus we can summarize the individual challenges that we have devised so far into a simpler and yet tougher grand challenge:

The provision of S&D in AmI ecosystems requires the dynamic application of the expertise of security engineers in order to dynamically react to unpredictable and ever-changing contexts. The intuitive solution would be to create an "intelligent" system able to analyze the requirements and the context in order to synthesize new solutions. Unfortunately, given the state of the art in both security engineering and intelligent systems, this approach is not a promising one in the foreseeable future. To meet this challenge in our time we need to look more closely to what technology is available for S&D mechanism in AmI ecosystems.

Security related common problems of systems can be classified into three categories, according to whether they threat confidentiality, integrity or availability of systems. In the following we will describe each of them.

Confidentially is the property that information holds when it remains unknown to unauthorized principals. In the case of AmI environments almost all the communication are carried out through wireless connections. It is well known that wireless connections are more vulnerable to attacks than wired connections since the information could be transmitted to anyone in the network range. Hence, we would expect that some of the security solutions existing for wireless networks could be adapted to AmI environments.

Among the most used techniques for distributed systems it is worth to mention those based on encryption and decryption. These techniques achieve a certain level of security by obscurity. Examples of these techniques are stream cipher and; block cipher techniques by Vernam and Maubogne (Richard W. Hamming, 1980) More recent techniques in the same line of research include those introduced in (Gideon Yuval, 1979) by Schneier. Public Key Infrastructure is also a technique used to achieve confidentiality. In some cases, it is more convenient to combine both public and private key cryptographic systems. This leads to the well known hybrid systems.

Integrity is the property that is violated when information is altered without authorization. This definition applies to the information held in a host as well as for the information in transit between hosts. As we mention before, wireless networks are more vulnerable to attacks than wired ones. The main reason being that anyone on the range of the wireless network can receive the signal. Thus, a man-in-the-middle attack is easy to be performed and, as a consequence, an attack on the data integrity.

Some of techniques used widely in distributed systems are Errors Detection Code (Gideon Yuval, 1979), Hash's tables (Maña et. al, 2006), MAC (Message Authentication Code) or Digital Signature. These techniques could be also applied to the new emerging environments of AmI, though we should take into account the new problems arising, concerning the nature of AmI.

Availability is the property of a system that grants and legitimizes requests by the authorized parties. A possible attack occurs when a malicious principal is able to achieve that the service is denied to an authorized principal by means of overloading the system. In any AmI environment where the users are connected to the host, it is possible to carry out an attack by denying the service. This could put under risk the availability of the system.

4 INTRODUCING THE SERENITY APPROACH

The objective of the SERENITY Project is to provide a framework for the automated treatment of security and dependability issues in AmI scenarios. In order to do that the SERENITY Project focuses has two main cornerstones: (i) capturing the specific expertise of security engineers in a way that allows its automated processing; and (ii) providing means to perform run-time monitoring of security and dependability mechanisms. They have been deployed by means of:

1. A set of modelling artefacts used to capture security expertise (called S&D artefacts). We use the term S&D solution to refer to an isolated component that provides a security and/or dependability service to an application. S&D artefacts are used to represent S&D solutions at different levels of abstraction. The representation of S&D solutions at different levels of abstraction responds to the needs of using them at the different phases of the software development process. These S&D artefacts are presented in this section.
2. A development-time framework (the SERENITY Development-time Framework, SDF) supporting:
   - The development of S&D solutions by means of the aforementioned S&D artefacts. The SDF includes processes and tools used by security experts for the creation of new S&D solutions. S&D solutions developed using the SDF include semantic information related to its informational description and its operational behaviour.
   - The development of secure applications following the SERENITY approach. These secure applications rely on SERENITY for fulfilling its security and dependability requirements. Applications developed by this way are called SERENITY-aware applications. They include references to SERENITY S&D artefacts. At run-time these references are resolved so that S&D solutions are provided and can be used by applications.

   The SDF is supported by on-line repositories populated with S&D artefacts. Security experts use these on-line repositories in order to store the S&D solutions they develop. And, application developers access to on-line repositories when they are developing SERENITY-aware applications in order to look for S&D solutions to reference from their applications. A detailed description of the development of application based on these on-line repositories can be found at (Serrano et. al, 2008).
3. A Run-time framework, called SERENITY Run-time Framework (SRF). The SRF provides support to applications at run-time, by managing S&D solutions and monitoring the systems' context. SERENITY-aware applications are developed by means of open architectures that are complemented at run-time by the SRF.

In order to facilitate the understanding of the SERENITY approach the rest of this section introduces how it is possible to capture security expertise by means of the SERENITY S&D artefacts.

The SERENITY Project provides five main S&D artefacts to represent S&D solutions: S&D Classes, S&D Patterns, Integration Schemes, S&D Implementations and Executable Components. These S&D artefacts, depicted in figure 1, represent S&D solutions using semantic descriptions at different levels of abstraction. All S&D artefacts, but Executable Components, are represented using XML files. Executable Components are code.

- To start, S&D Classes represent abstractions of a security service that can be provided by different S&D solutions characterized for providing the same S&D Properties and complying with a common interface. At the level of S&D Classes, S&D solution descriptions are very simple, containing some information about the name of the solution, and its creators, the security properties provided, and the interface offered by the solution. Regarding to the security properties, SERENITY provides a formalism created for representing and reasoning about security properties (called S&D properties), interested readers can refer to (Aresdani Aboba, 2003) .
- S&D Patterns are detailed descriptions of abstract S&D solutions. As presented in Figure 1, each S&D Pattern belongs at least to one S&D Class. At this level of abstraction the description of S&D solutions are more detailed than in the

previous one (S&D Classes). These descriptions contain all the information necessary for the selection, instantiation, adaptation, and dynamic application of the security solution represented in the S&D Pattern. Since each S&D Pattern may have a different interface, they contain a specification that allows mapping the abstract calls defined in the S&D Class interface into the specific calls defined by the S&D Pattern interface. Also, S&D Patterns include behavioural description of the security mechanisms they represent. Besides, they include the pattern semantics, which are related to the semantics of the security properties provided. Finally, S&D Patterns include information about the restrictions imposed by the solution. It is important to take into account that the S&D Patterns we are using in the SERENITY Project differ from the current concept of patterns in software engineering. S&D Patterns are components containing detailed information about S&D solutions. These components (S&D Patterns) are machine readable.

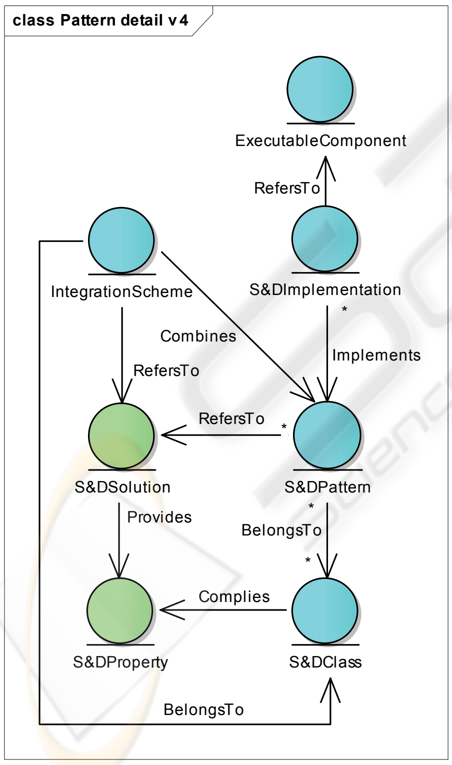

Figure 1: Class Pattern detail.

- Integration Schemes are an especial type of S&D Pattern that represent S&D solutions that are built by combining several S&D Patterns. While S&D Patterns are independent or atomic descriptions of S&D solutions, Integration Schemes describe solutions for complex requirements achieved by the combination of smaller S&D solutions (represented by means of S&D Patterns).
- S&D Implementations represent the components that realize the S&D solutions. S&D Implementations are not real implementations but their representation/description. This is the lower abstraction level, at this level S&D solutions are defined in terms of the technology used in their development and how to make use of it.
- Finally, Executable Components are the actual implementations (pieces of code) of S&D solutions. The automatic processing of Executable Components is based on the use of the information provided by the S&D artefacts representing the S&D solution implemented by them.

S&D Classes, S&D Patterns and S&D Implementations are development-time oriented artefacts, and Executable Components are especially suitable for run-time. These S&D artefacts are organized as a hierarchy, that is to say, each S&D Class has several S&D Patterns, and each S&D Pattern has several S&D Implementations. As aforementioned, SERENITY-aware applications include references to S&D artefacts. Depending on the S&D artefact level of abstraction used by application developers, at run-time the SRF is more flexible when selecting S&D Solutions. The main purpose of introducing this hierarchy is to facilitate the dynamic substitution of the S&D Solutions at run-time, while facilitating the development process. For instance, at run-time all S&D Patterns (and their respective S&D Implementations, see figure 1) belonging to an S&D Class will be selectable by the SRF in response to an S&D Class-based request by the application.

5 THE PROOF OF CONCEPT: A WIRELESS COMMUNICATIONS PROTOTYPE SCENARIO

In this section we describe a target scenario for applying previously presented SERENITY approach, as well as the corresponding prototype from a SERENITY perspective, as a remarkable example of a scenario related on AmI and with direct application in the industry. It is important to clarify that we can't consider any pure AmI scenario for the time being, since AmI is an emerging concept and current technologies don't allow actual AmI deployments in the widest sense of the term. But he proposed scenario depicts key characterises of AmI environments. In fact, the scenario shows that in addition to the impact in the future AmI scenarios, the SERENITY approach can have a short-to-medium term impact as an enhancement of many current technologies.

The proposed scenario focuses on the provision of seamless and access-controlled communication over a wireless network, which provides connection and access to multiple resources in the company, such as internal digital documents, data bases, intranet services and internet connection. Moreover, it offers a convenient mobility to users, who do not need to be physically connected to any cable or access point.

In a conventional company network, security policies are usually assigned in a fixed way, thus not including AmI features or dynamic changes based on context at run-time, and a thorough knowledge of S&D issues is needed when deploying the wireless network and the associated access control to information. In order to improve this conventional situation, in the proposed scenario we include AmI features and use the SERENITY model. In this AmI communication scenario, several context features are considered in order to improve S&D. In particular, we highlight user location and device authentication.

Location is a key factor in the access control policies of the scenario: some re-sources may or may not be accessed depending on the current position of the re-questing user in the office. Location information is provided by an existing Indoor Location System (ILS). Furthermore, device identity is also used as a factor to allow/deny access to certain resources: only users whose devices are properly authenticated as company devices (and thus trusted devices), will have access to resources with a high security level.

In the scenario we consider several situations dealing with the network security. In all of them we have one or more users trying to access resources through the wireless network from a specific location. Users are first authenticated with their identity, associated with a user profile. An Access Control Server (ACS) decides to grant or deny the access taking into account:

- Location: using information provided by an ILS.
- User identity: users are authenticated to initiate a session and then can be properly classified as a certain profile (administrator, employee, visitor…).
- Device identity: each device is identified by using a Trusted Platform Module (TPM).

We can extract many requirements from this scenario, but for the sake of simplicity we focus on S&D requirements and within them especially on those related to Ami issues. In order to better address them, a SERENITY-enabled prototype has been developed, aiming to denote the benefits of applying the SERENITY approach and show its good suitability for AmI environments. Taking into account those S&D requirements, we have designed the following prototype architecture where the SERENITY model has been adopted:

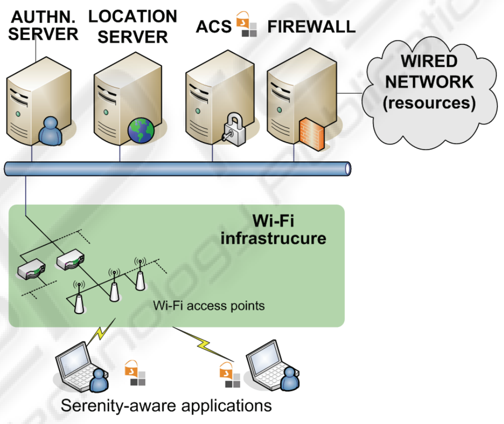

Figure 2: Scenario Architecture.

In this figure, we can recognize all the main actors involved in the scenario. At first glance you may observe the SERENITY enabled entities (denoted with the SERENITY icon). These entities include an SRF instance providing all the functionalities mentioned in section 4.

- The Wi-Fi infrastructure is main access network in the company. Additionally, it is the base for localization since it's achieved by Wi-Fi signal triangulation (using several access points).
- The Authentication Server processes user connection requests, and allows or denies them attending to presented user credentials. It uses a database as a user data repository. It uses the EAP-RADIUS protocol (Chiba et. al, 2008).
- The Location Server implements a Wi-Fi based ILS that tracks real-time location of all connected users.
- User devices can be any able to connect to Wi-Fi network. In our scenario we use laptops with TPM chips for device identification, as well as

SERENITY-aware client applications and a SRF instance.

- The ACS runs a Control Application which implements the core functionalities of the system: it controls the user accesses to resources based on user profiles, their location and the "identity" of their devices. This Control Application is designed a full SERENITY-aware application that relies on an instance of the SRF, which is responsible for the selection and provision of the most suitable S&D solutions to fulfils requests from the applications. For instance, the SRF decision may result in a dynamic reconfiguration of filter rules in the firewall.
- The Firewall isolates the wireless network from the rest of the (wired) network and is dynamically configured by ACS (rules can be changed "on the fly")

The prototype point out the good conditions of the SRF for AmI environments by showing its negotiation feature in action: there is one SRF instance in the central ACS and several instances in client devices. The central SRF and device instances can interact and negotiate with the SRFs in the devices in order to provide appropriate distributed solutions such as client-server protocols, enhancing distributed security and flexibility.

In the prototype we provide a set of artefacts addressing specific requirements for the scenario but these artefacts are not strictly bound to it. Taking advantage of the S&D Pattern approach, these artefacts could be used in other environments since they represent independent S&D Solutions. Alternatively, we might have used previously developed S&D patterns and solutions instead of developing them. The artefacts used for the scenario are:

- TPM-based device identification pattern: this pattern represents a mechanism to identify a device, based on the TPM technology. In the same way that humans identify themselves in different ways (e.g. by means of biometrics), a TPM-enabled device can claim its "identity" by cryptographic means, using a TPM for that purpose. TPMs provide a set of hardware-based cryptographic functions that allow making these claims in a trusted way. TPMs implement a challenge-response protocol that allows control servers to obtain proofs of the identity of the device.
- Zone-based security measurement pattern: the solution represented by this pattern provides a security assessment for specific zones inside controlled areas: obtain a qualitative value or measurement of the security taking in count predefined zone profiles.
- Access Control Integration Scheme: in our scenario, previous patterns are conceived to work together. The different solutions represented by these patterns in isolation are not enough to take a decision on access control. However, a combined solution can be used to provide fine-grained access control. For this purpose, SERENITY provides a very useful artefact: the Integration Scheme that allows the creation of a new pattern based on the composition of other patterns. In our case an Integration Scheme is used to provide a final access control decision.

The overall result of the application of SERENIY approach on the proposed scenario shall be denoted by the following perceived benefits: Simplicity for adding new factors to the control access, that is the system administrator just needs to add new patterns implementing those factors to de S&D Library. Combining the factors is as easy as creating Integration Schemes using different patterns. And last but not least the high adaptability provided by the event collector mechanism and the monitoring system the control system counts with a ready-to-use high adaptable engine to provide the best solution available in each case.

On the other hand, we must not ignore some implications and challenges. Firstly extra code needs to be added in order to get a proper Executable Component, and therefore to be executed in a SRF environment. At this point, dealing with proprietary software presents an important issue: in the real world, system developers and integrators use existing commercial solutions, which normally can't be modified. In this case we have used software wrappers providing the SERENITY interface. Providing the events and monitoring data, an extension of the previous point is considering all the monitoring stuff you need to provide to allow SRF to treat correctly the use of every S&D Solution. However, these issues must not discourage potential adopters of SERENITY.

6 CONCLUSIONS

The realization of the Ambient Intelligence concept entails many important challenges, but the most important barriers to this realization, is the lack of adequate support for security. Along this paper we have presented the SERENITY approach consisting on a new model for addressing the security issues in the development of distributed applications and systems for Ambient Intelligence scenarios. We showed as the main objective of the SERENITY Project is to provide a framework for the automated treatment of security and dependability issues in AmI scenarios. SERENITY is focused on two main cornerstones: (i) capturing the specific expertise of security engineers; and (ii) providing means to perform run-time monitoring of the security and dependability mechanisms. Finally we provide a proof of concept by means of the Wireless Communication prototype, which is related on AmI and with direct application in the industry.

ACKNOWLEDGEMENTS

This work is funded by the *6th Framework Programme of the European Union* developed in the SERENITY (IST-027587) Project.

REFERENCES

SERENITY project. Funded by European Commission. Directorate General Information Society & Media.Unit D4 - *ICT for Trust and Security*, under grant IST-027587. http://www.SERENITY-project.org, 2006.

Merabti M. Shi Q. Askwith B. Llewellyn-Jones, D. Utilising component composition for secure ubiquitous computing. *In Proceedings of 2nd UK-UbiNet Work-shop.*, 2004.

Zhang Shi, Q. An effective model for composition of secure systems. 1998. *Journal of Systems and Software*, 433:233-44.

BEA White Paper. *BEA WebLogic Security Framework: Working with Your Security Eco-System.* http://www.bea.com.

Object Management Group. *The Common Object Request Broker: Architecture and Specification.* http://www.omg.org.

Gilson Wilson and Ullas O. Tharakan. Unified security framework. In Trinity College Dublin, editor, *In ISICT '03: Proceedings of the 1st international symposium on Information and communication technologies,* pages 500-505, 2003.

Le Gruenwald Carlos Sanchez and Mauricio Sanchez. A monte carlo framework to evaluate context based security policies in pervasive mobile environments. In New York, USA, ACM, editor, *In MobiDE '07: Proceedings of the 6th ACM international workshop on Data engineering for wireless and mobile access.,* pages 41-48. ACM, 2007.

R. Sampemane G. Ranganathan A. Campbell R.H. Hill. A framework for automatically satisfying security requirements. *In Workshop on pecification and Automated Processing of Security Requirements' - SAPS'04 at the 19th IEEE International Conference on Automated Software Engineering.*, 2004.

C. Boudaoud, K.; McCathieNevile. An intelligent agent-based model for security management. *In iscc, editor, Seventh International Symposium on computers and Communications*, page 877, 2002.

IBM's Security Strategy team, 2004. *Introduction to Business Security Patterns.* An IBM White Paper. Available at http://www-3.ibm.com/security/patterns/intro.pdf. 2004.

J. Yoder and J. Barcalow. Architectural patterns for enabling application security. *In MA: AddisonWesley Publishing Company.* Reading, editor, Pattern Languages of Program Design., volume 4, pages 301-336, 2000.

E.B. Fernandez. Metadata and authorization patterns. *In Technical report, Florida Atlantic University, 2000.*

Romanosky, S., 2001. Security Design Patterns, Part 1, 1.4.

E.B. Fernandez and Rouyi. Pan. A pattern language for security models. *In PLoP01 Conference., 2001.*

Soundarajan N. Hallstrom, J. O. Pattern-based system evolution: A case-study. *In the Proc of the 18th International Conference on Software Engineering and Knowledge Engineering. San Francisco Bay, USA., 2006.*

T. Mikkonen. Formalizing design patterns. In IEEE Computer Society Press.,editor, *In Proc. Of 20th ICSE.*, pages 115-124, 1998.

Richard W. Hamming. *Coding and Information Theory.* Prentice-Hall, 1980. ISBN 0-13-139139-9.

Gideon Yuval. "*How to Swindle Rabin*". Cryptologia, 3: 187189, Jul 1979. ISSN 0161-1194.

A. Maña, C. Rudolph, G. Spanoudakis, V. Lotz, F. Massacci, M. Melideo, and J. M. López-Cobo. *Security Engineering for Ambient Intelligence: A Manifesto.* Integrating Security and Software Engineering. IDEA Group, 2006. ISBN 1-59904-148-0.

Daniel Serrano, Antonio Maña, and Athanasios-Dimitrios Sotirious. Towards precise and certified security patterns. *In Proceedings of 2nd International Workshop on Secure systems methodologies using patterns, Spattern 2008,* pages 287-291, Turin, Italy, September 2008. IEEE Computer Society. ISBN 978-0-7695-3299-8.

Aresdani Aboba B, Calhoun P, 2003 *RADIUS* Support For Extensible Authentication Protocol EAP. IETF RFC 3579 , updates: RFC 2869.

Chiba M, Dommety G, Eklund M, Mitton D, Aboba B., 2008 *Dynamic Authorization Extensions to Remote Authentication Dial In User Service.* IETF RFC 5176. Obsoletes: RFC 3576.