A Privacy Enhancing Architecture for Collaborative Working Environments

Jasone Astorga, Purificacion Saiz, Eduardo Jacob, and Jon Matias

University of the Basque Country
Faculty of Engineering. Alameda de Urquijo s/n. 48013 – Bilbao
{jasone.astorga,puri.saiz,Eduardo.jacob,jon.matias}@ehu.es

Abstract. Recent widespread deployment of different types of sensors and detectors has opened the door to a new way of understanding collaborative applications. The use of such devices allows information to be collected, used and disclosed on a massive scale and under very different conditions from which we are currently familiar with. Despite the huge potential of these collaborative environments, privacy is one of their most criticized aspects and probably the greatest barrier to their long-term success. To address this problem, we present a privacy-enhancing security model specifically tailored to the characteristics of ubiquitous and heterogeneous environments consisting of low capacity devices. This security model is based on the Kerberos symmetric key protocol which has been modified and extended to avoid using timestamps and to accomplish the authorization process. Finally, we present a validation of our proposal using an automated tool and we show a real world deployment use case.

Keywords: Kerberos, Centralized Authorization, Privacy, Sensors.

1 Introduction

Companies and organizations have traditionally promoted collaboration as a way of improving their processes and ultimately saving time and money. Frequently, the used collaborative tools rely on distributed applications in which different devices and software modules interact with each other. The resulting collaborative environments often integrate mobile devices such as PDAs or laptops to provide users with ubiquitous access to the system, but also sensors and other low capacity devices, which are used to collect data or provide real-time information. This gives place to a new trend of smart environments, mainly characterized by their invisibility and pervasiveness. Despite their huge potential value, these kinds of environments must still deal with some key challenges, being privacy one of the most important ones. In fact, this type of system dreadfully raises the level of the challenge to protect end-users' privacy, mainly due to the unprecedented data collection coverage, the invisibility of the collection process, the amount of data collected and the envisioned system connectivity. In this regard, we understand the concept of privacy as defined by Westin as "the claim of individuals, groups or institutions to determine for themselves when, how, and to what extent information about them is communicated to others" [1].

The aim of our work is to develop an infrastructure that allows the construction of privacy-aware collaborative applications integrating low capacity devices. With this purpose, we have developed a security model which allows the enforcement of different privacy enhancing mechanisms such as authentication and authorization of remote parties and secrecy of private data. In order to address the authentication issue, this model relies on the standard Kerberos [2] protocol. However, we propose a modification of this protocol so that it also provides authorization functionalities. Additionally, after a successful authentication and authorization step, communicating parties share a secret key which can be used to implement additional functionalities such as ensuring the integrity and confidentiality of the transmitted data.

2 Collaborative System Architecture

The considered collaborative application model is based on a layered architecture that follows the principles of an Open Service Oriented Architecture (OSOA). This multi-layer architecture relies on decoupled building blocks that deal with different aggregation levels of business functionality, namely *core services,* which provide specific functionalities, and higher lever *software collaborative tools*. As shown in Fig. 1, the core services are defined as reusable software modules that implement very diverse basic or core functionalities, while the software collaborative tools are defined at a layer above. Thus, they can exploit one or more core services to offer aggregated functionalities.

Given the distributed and dynamic nature of the envisioned collaborative applications, such a system cannot be conceived without reliable mechanisms to ensure certainty in peer identification, to restrict data dissemination to the desired entities, and to guarantee the secrecy and integrity of these communications. Although these are common security tasks, a new level of difficulty appears when it comes to dealing with highly dynamic and radically unpredictable elements and interactions.

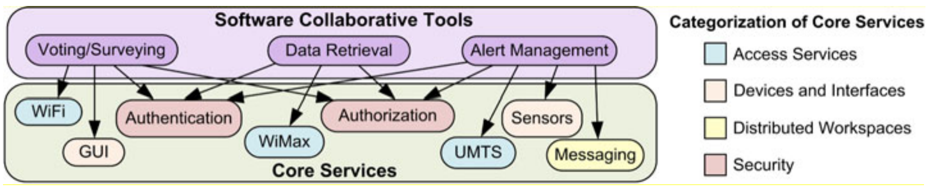


Fig. 1. Examples of collaborative applications based on the layered architecture model

2.1 Privacy and Security Concerns

Privacy and *security* are very closely related concepts. *Privacy* implies the possession of some kind of information and the subsequent terms and conditions by which it may be used, retained and disclosed to others, while *security* is often used to describe the capability of a technical system to protect and maintain the privacy of the information circulating within that system. Therefore, privacy breaches can occur when a system is not secure and leaks information to unauthorized parties.

From the data exchange point of view, building a privacy-aware collaborative architecture entails the implementation of security mechanisms to provide authentication and authorization of remote peers as well as to ensure the integrity and confidentiality of the transmitted information. This invariably results in the utilization of cryptographic algorithms, which are by nature highly resource consuming. However, typical small information gathering devices are mainly characterized for having small batteries and limited processing and storage capacity. Thus traditional security mechanisms are not directly applicable to these environments; and more specifically the use of asymmetric cryptographic algorithms is not preferable as they are more resource intensive than symmetric key algorithms [3].

From the logical point of view, one of the key concepts of the collaborative architecture introduced above is that the core services must be neutral and independent from the higher level applications that make use of them. As a result, it is not feasible for the core services to maintain and manage identity and authorization related information regarding all the remote parties they may interact with. Additionally, trust relationships among the same core services may vary from one collaborative application to another. For these reasons, a centralized management of identity and access rights related information is essential to ensure the simplicity and scalability of the system.

3 Proposed Solution

The aim of our work is to provide the above introduced architecture with mechanisms to ensure that the communications between the different entities that compose the collaborative applications are authenticated, authorized and protected from eavesdropping and modification by third parties. To that end, we have designed a security protocol which deals with the two major constraints of the considered environments: (1) the resource limitations of the involved devices make it necessary to keep the communication overhead as well as the computation power to a minimum, and (2) the dynamic nature of the collaborative applications makes a centralized management of authentication and authorization processes preferable.

Following these design principles, we have developed a lightweight security model based on the use of symmetric key cryptography, and more specifically, on the Kerberos authentication protocol. However, we have enhanced this protocol so that apart from authenticating the identities of the requesting entities, it also verifies their access rights. This way, all the information regarding identities and rights is maintained in a centralized location, where it can be easily accessed and updated by a system administrator. Additionally, the actual authentication and authorization processes are executed out of the basic components, relieving these entities of time and computation power consuming tasks, such as exhaustive searches, etc. Regarding the previously introduced distributed architecture, the security modules are integrated into it as core services, supporting this way, different higher level collaborative applications.

A detailed description of the underlying cryptographic mechanisms of this privacy enhancing model is beyond the scope of this paper; readers desiring a thorough review of these mechanisms are referred to [4].

3.1 Why a Kerberos-Based Approach

Kerberos [2] is a time-tested, widely-deployed system that provides authentication and the establishment of secure channels in open networks. As it is a well-known protocol, it will not be explained in detail here, but it is worth reminding some of its terminology and basic operation concepts, as they will be used later in this paper.

Each client or service is known as a *principal* in Kerberos, and each principal is characterized by owning a secret key known only by the principal itself and the Kerberos Key Distribution Center (KDC). The Kerberos authentication mechanism is based on the use of *tickets*. A *ticket* is a capability distributed by the Kerberos KDC that contains a proof of the identity of the principal that requested it. The tickets are encrypted so that only the entities for which they are intended are able to decrypt them. Therefore, each client that wants to authenticate to a server will present a ticket issued by the Kerberos KDC for that service.

Some of the benefits of Kerberos that make it a suitable technology for our approach are that it prevents the transmission of passwords over the network and makes use of a centralized user account administration. However, Kerberos also presents some constraints that make its deployment difficult: it requires synchronization between the participants and it lacks an authorization service.

3.2 Related Work

Most of the work carried out so far regarding security in sensor networks [5], [6] focuses on protecting the communications among these devices at the physical or MAC layer, but they are not suitable to manage security at the application level.

On the other hand, there have been numerous efforts to add authorization support to Kerberos, being the most remarkable ones SESAME [7], IDfusion [8], an implementation based on restricted proxies [9] and Microsoft's implementation of Kerberos [10]. We have studied all these protocols and concluded that they all present different drawbacks which make them unsuitable for the environments considered in this work, being the most remarkable ones the use of public key technology in different steps and the lack of a centralized management of authorization information.

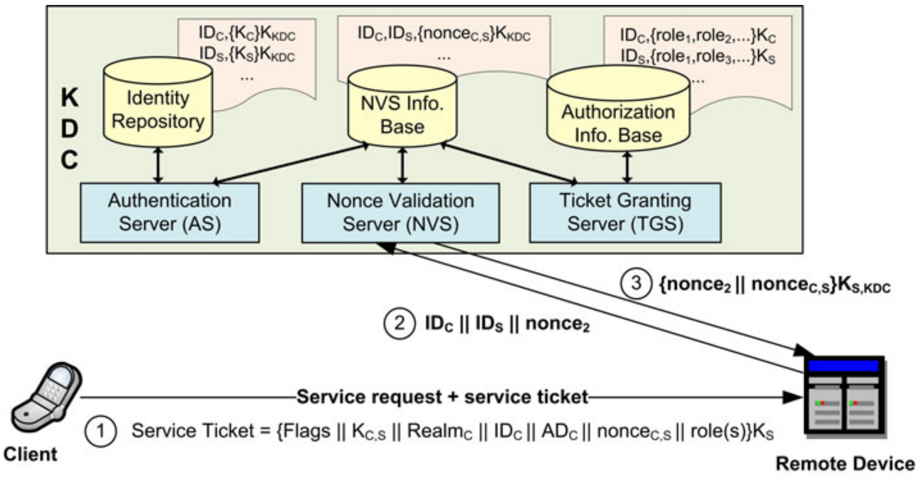
3.3 The Time Synchronization Issue

Kerberos makes use of timestamps as a way of proving the freshness of the messages, and thus avoiding reply attacks. One major drawback of using timestamps is that it requires synchronization among all the interacting entities. However it also presents some desirable properties, such as statelessness, which is extremely valuable from the scalability point of view.

In order to avoid the necessity for synchronized clocks, we propose a *nonce*-based implementation of Kerberos, which basically uses the *authtime* field of the Kerberos tickets and protocol messages to include a nonce value. To provide the participating entities with a mechanism to check the validity of these nonce values, we have introduced the concept of a *Nonce Validation Service (NVS)*. The NVS is a new service that resides in the Kerberos Key Distribution Center (KDC), together with the Authentication Server (AS) and the Ticket Granting Server (TGS). As a result, the developed system becomes stateful, but it has the advantage that the state-related information is only maintained in the KDC and not in the end nodes, minimizing this way its impact on the system's scalability.

3.4 The Authorization Issue

The architecture of the proposed security protocol is shown in Fig. 2, which deviates from the conventional Kerberos protocol essentially in the introduction of two new information stores and the extension of the protocol with two new messages. Regarding the new information stores, one is used by the NVS to store the nonce values associated to each ticket, and the other contains information regarding access control restrictions, based on a role-based access control (RBAC) model.



$ID_C,\{K_C\}K_{KDC}$
$ID_S,\{K_S\}K_{KDC}$
...
$ID_C,ID_S,\{nonce_{C,S}\}K_{KDC}$
...
$ID_C,\{role_1,role_2,...\}K_C$
$ID_S,\{role_1,role_3,...\}K_S$
...
KDC
Identity Repository
NVS Info. Base
Authorization Info. Base
Authentication Server (AS)
Nonce Validation Server (NVS)
Ticket Granting Server (TGS)
3 $\{nonce_2 \| nonce_{C,S}\}K_{S,KDC}$
2 $ID_C \| ID_S \| nonce_2$
Service request + service ticket
1 Service Ticket = $\{Flags \| K_{C,S} \| Realm_C \| ID_C \| AD_C \| nonce_{C,S} \| role(s)\}K_S$
Client
Remote Device

Fig. 2. Basic architecture and message exchanges of the proposed security protocol

Kerberos provides a user with the necessary credentials to access a service in two steps: first, it authenticates the user and provides him with the necessary elements to establish secure and authenticated communications with the KDC. Then, it generates specific credentials for each service the user wants to communicate with. It is before the second step where the authorization rules are enforced, limiting this way the generation of authentication credentials to legitimate authorized users. That is, whenever the KDC receives a request for a Service Ticket, before generating it, it verifies if the requesting principal owns the necessary rights to access the desired end service. With this purpose, the KDC queries its local authorization base and compares the roles with access right to the service with the roles that can be undertaken by the user. If a match is found, the user is determined to be authorized to access the desired service and the requested Service Ticket is generated, embedding in its authorization payload field the role undertaken by the client.

When a resource provider receives a new service request accompanied by a Service Ticket, it has to validate the content and the format of the received ticket. The validity of this ticket asserts that the identity claimed by the client is true and also that the client is authorized to access the given service. Therefore, the resource providers rely on the Kerberos KDC for both authentication of remote users and control of unauthorized accesses. The validation of a Service Ticket consists basically of three steps. First the ticket must be successfully decrypted with the service principal's secret key. Second, the validity of the nonce value contained in the ticket must be checked against the Kerberos KDC, for which a new request/response message pair has been introduced in Kerberos to allow querying the NVS, as shown in Fig. 2. Third, it must be guaranteed that the received Service Ticket contains a role identifier embedded in its authorization payload field.

Finally, it must be noted that the generation of a Service Ticket involves some resource consumption: bandwidth consumption, as the ticket must be transmitted from the KDC to the requesting entity and then from this entity, to the desired end server; and CPU consumption, mainly in the resource provider, as it has to decrypt the received Service Ticket and perform the validation of the incoming request. All these processes become a waste of resources when it can be determined beforehand that in the end the service will not be provided because the user is not authorized to access it. Therefore, limiting the generation of Service Tickets to legitimate requests reduces the network load, as well as the data processing performed by the service principals.

4 Formal Validation of the Proposed Security Protocol

As security protocols are notoriously difficult to design and extremely error-prone, new security approaches must be validated before deployment. Taking into account that with informal reasoning it is hard to consider all possible actions that an adversary may perform to breach security, formal verification has proved to be an essential tool in the verification of security protocols. The tool we have used to analyze our protocol has been AVISPA: Automated Validation of Internet Security Protocols and Applications [11]. This tool relies on the HLPSL (High Level Protocol Specification Language) protocol for the formal specification of the security protocol to be assessed and it integrates with four different back-ends, which perform the actual analysis of the protocol. All back-ends assume perfect cryptography, that is, an attacker cannot solve encryption without the knowledge of the whole key. Additionally, the communication channels are based on a Dolev-Yao intruder model, which means that the attacker has basically full control over the channel.

An essential part of the HLPSL description are the *security goals*, as the security analysis is performed against these goals and the results indicate whether the protocol meets them or not. AVISPA provides templates for two of the most frequently used security goals: *authentication* and *secrecy*. Among the properties that a security protocol may satisfy we have defined the following as critical to ensure our protocol's reliability and modelled them accordingly:

- *Authentication*: the protocol should be able to check and provide guarantees about the identity of any component of the system. This property is modelled by the *authentication_on* and *weak_authentication_on* security goals.
- *Access control*: the protocol should avoid the unauthorized use of resources or access to data. This feature is assessed by checking that every Service Ticket contains at least one role_ID in its authorization payload field and by enforcing the *secrecy_of* security goal on this field.

- *Data confidentiality and data integrity*: the protocol should provide protection against the disclosure and modification of the data during a communication. These features are validated by enforcing the *secrecy_of* security goal on all the exchanged secret or session keys.

One crucial aspect of a security validation is the initial knowledge allocated to the intruder. In this sense, first we have validated the protocol by implementing a single session and allowing the intruder to play the role of each legitimate agent. Then we have evaluated the case in which two parallel sessions of the protocol are executed, and in one of them, one of the legitimate agents is playing a role for which it is not intended to. AVISPA does not report about any attack in any of the cases.

5 Architecture Deployment in a Real Environment

The security model introduced in this paper has been developed and implemented under the scope of the C@R project, an Integrated Project under the 6th Framework Programme. The aim of C@R (A Collaborative Platform for Working and Living in Rural Areas) is to promote collaborative environments in rural areas in order to enable their development and permit their integration in the information society. To achieve this goal, a novel architecture for the composition of collaborative applications has been developed, in which the introduced security model has been integrated. Then, this architecture has been validated following a Living Lab methodology [12]. As a specific example, the Cudillero Living Lab [13] has implemented a use case with the objective of obtaining a quality hallmark with origin certificates for hake catches.

To implement this use case both fishermen and fishing boats should be equipped with different types of sensors providing information about location, temperature, humidity, etc. Regarding the origin certification of the catches, location information is a critical piece of data. Nevertheless, this information is also highly sensitive, as fishermen are not willing to reveal their fisheries to potential competitors, and thus in normal conditions it must be kept secret and only available to entitled parties. However, this fact changes dramatically when an emergency situation arises. In this case location information must be automatically disclosed to emergency services and also to any other boat nearby. Therefore, the deployed security model must be able to deal with the dynamic and real-time management of access rights.

6 Conclusions

The aim of this paper has been to draw attention to the privacy needs that must be addressed so that the deployment of dynamically built collaborative applications is feasible in environments involving low capacity devices. For this purpose, the basic characteristics of the targeted application scenarios have been studied, identifying the privacy and security issues affecting them. These issues have led to the high-level requirements of (1) a lightweight cryptographic solution and (2) a centralized management of authentication and authorization related information.

The introduced security model meets these requirements allowing the different entities that compose a collaborative application to establish trust relationships for the secure exchange of data. Although our model provides the necessary cryptographic material to protect these data, it does not specify whether they should be protected or not, or in which manner. This will in fact depend on the specific collaborative application above and on the critical nature of the transmitted information.

References

1. Westin, A.F.: Privacy and Feedom. Atheneum, New York (1967)
2. Neuman, C., Hartman, S., Raeburn, K.: The Kerberos network authentication service, v5 (2005), `http://www.ietf.org/rfc/rfc4120.txt`
3. Ruangchaijatupon, N., Krishnamurthy, P.: Encryption and power consumption in wireless LANs. In: 3rd IEEE Workshop on Wireless LANs, Newton, Massachusetts (2001)
4. Astorga, J., Matias, J., Saiz, P., Jacob, E.: Security for Heterogeneous and Ubiquitous Environments Consisting of Resource-Limited Devices: An Approach to Authorization Using Kerberos. LNICST, vol. 42, pp. 65–76. Springer, Heidelberg (2010)
5. Perrig, A., Szewczyk, R., Tygar, J.D., Wen, V., Culler, D.E.: SPINS: security protocols for sensor networks. ACM Wireless Networks 8(5), 521–534 (2002)
6. Karlof, C., Sastry, N., Wagner, D.: TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. In: 2nd International Conference on Embedded Networked Sensor Systems (SenSys 2004), Baltimore, MD, USA (2004)
7. Kaijser, P., Parker, T., Pinkas, D.: SESAME: the solution to security for open distributed systems. Computer Communications 17(7), 501–518 (1994)
8. Wettstein, G.H., Grosen, J.: IDfusion, an open-architecture for Kerberos based authorization. In: AFS and Kerberos Best Practices Workshop, Michigan (2006)
9. Neuman, C.: Proxy-based authorization and accounting for distributed systems. In: 13th International Conference on Distributed Computing Systems, Pittsburgh, pp. 283–291 (1993)
10. Walla, M.: Kerberos explained, issue of Windows 2000 Advantage magazine (2000), `http://technet.microsoft.com/en-us/library/bb742516.aspx`
11. AVISPA: Automated Validation of Internet Security Protocols and Applications. FET Open Project IST-2001-39252 (2003), `http://www.avispa-project.org`
12. Schumacher, J., Feurstein, K.: Living labs – a new multi-stakeholder approach to user integration. In: 3rd International Conference on Interoperability of Enterprise Systems and Applications (I-ESA 2007), Funchal, Portugal (2007)
13. Valenzuela, M., Sierra de Miguel, A., Navarro, M.M.: A Living Lab for Stimulating Innovation in the Fishery Sector in Spain. In: Schaffers, H., García, J., Navarro, M., Merz, C. (eds.) Living Labs for Rural Development. Results from the C@R Integrated Project, pp. 83–104. TRAGSA, Madrid (2010)