Security for Heterogeneous and Ubiquitous Environments Consisting of Resource-Limited Devices: An Approach to Authorization Using Kerberos

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

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

Abstract. Recent widespread of small electronic devices with a low capacity microprocessor and wireless communication capabilities integrated, has given place to the emergence of new communication scenarios, mainly characterized by their heterogeneity and ubiquity. As an example, in the near future, it will be very common for users to access and control electrical appliances or high performance sensors in remote locations just by using their mobile phone or PDA. However, for these environments to achieve the expected success they must probe to be secure and reliable. The security algorithms and mechanisms used to date are meant for powerful workstations and not suitable for small devices with specific constraints regarding energy and processing power. Therefore, in this paper we present a lightweight authentication and authorization solution based on the Kerberos symmetric key protocol, and we propose an extension of its functionalities in order to add authorization support.

Keywords: Authorization system, Communication system security, Kerberos.

1 Introduction

With the latest advances in semiconductor and electronic technologies and the affordability of Internet services, it is easier every day to find different electronic devices with a small microprocessor and wireless capabilities, such as Wi-Fi, Bluetooth, WiMax, ZigBee and UWB integrated. This is the case of different types of high performance sensors, electronic boards, handhelds, and even electrical appliances. Usually, these systems are characterized by small memories, limited computation power and by relying on batteries for their operation [1]. These characteristics are of vital importance when proposing a security model suitable for environments integrating this kind of devices.

An example of the considered scenarios could be the case of a user wanting to access different electronic devices or sensors at his or her home or office. For instance, in a hot summer day the user may want to connect to his or her air conditioning equipment so that when he or she arrives home the house is cool. On another occasion, the user may want to query the mobility sensor at his or her office in order to guarantee that it is currently free. Additionally, the given user could perform any of

Q. Gu, W. Zang, and M. Yu (Eds.): SEWCN 2009, LNICST 42, pp. 65–76, 2010.

these tasks from his or her mobile phone or PDA. A scenario with these characteristics can not be conceived without mechanisms that provide security. Actually, in such a case, instruments and sensors essentially become wireless physical information servers. Therefore, they must be protected in the same way as critical servers containing sensitive information or hosting vital business processes. In this sense, reliable authentication and authorization mechanisms are essential in order to avoid undesired actions to be performed by unauthenticated or unauthorized users.

A security mechanism suitable for these environments should minimize communication overhead and computation power. With these constraints it is impractical to use traditional security algorithms and mechanisms meant for powerful workstations. As a specific example, it is not practical to use asymmetric cryptosystems, and thus, key management protocols for these networks are based upon symmetric key algorithms.

Currently, the most common way to communicate with sensors or low capacity devices is by the deployment of proxies [2], [3]. The proxy is able to communicate with the sensor or low capacity device using some suitable protocol and act on behalf of this device when communicating with the external network. However, this solution presents several drawbacks: first, it is necessary to implement secure communication protocols between the end node and the proxy, and delegate the end node's credentials to the proxy so that it can act on behalf of the device. Besides this, the introduction of a new processing step in the communication penalizes the performance of the whole system. Another important drawback is that a proxy is usually designed to perform a specific task or implement a particular protocol, so such an implementation would require the development of special proxies for each application. On the other hand, recently, the efforts to introduce TCP/IP in sensors and low capacity devices are increasing considerably [4], [5].

In this paper we present a lightweight security model that provides authentication and authorization facilities in such ubiquitous and heterogeneous scenarios. For carrying out authentication this model relies on the standard Kerberos [6] symmetric key protocol. However, we propose a modification of this protocol so that it also provides authorization functionalities.

The rest of the paper is organized as follows. Section 2 provides a brief overview of the characteristics of the Kerberos protocol to show why it is a desirable mechanism for our approach and which the deficiencies that should be circumvented are. In section 3 we present an overview of different technologies that have been proposed in order to address the same issues, while in section 4 we provide a detailed description of our approach. Finally section 5 concludes the paper.

2 Kerberos-Based Approach

Kerberos is a time-tested, widely-deployed system that provides authentication and the establishment of secure channels in open networks. Each client or service in an administrative domain is called 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. Kerberos includes mechanisms to prevent forgery of client or server identity, detect reply attacks, distribute temporary session keys for the establishment of secure channels, etc. Readers desiring a review of the Kerberos protocol are referred to [6].

Among the benefits of Kerberos that make it a suitable technology for our approach are that it prevents the transmission of passwords over the network, provides Single Sign-On functionalities and makes use of a centralized user account administration. However, Kerberos also presents some constraints that make its deployment difficult: it presents synchronization problems, due to the fact that the system clocks of all the entities that make use of Kerberos must be synchronized; and it lacks an authorization service.

The Kerberos protocol provides the mechanisms to authenticate the identity of a client to a service, but it does not provide any information about the rights of the client to access the requested service. In this situation, it is the service itself who must store and manage the privileges of the client, and implement the corresponding access control mechanisms. This fact involves also a scalability problem, because in a distributed environment consisting of different networks interconnected among them, it is not feasible that every server stores information about the privileges of each and every possible client. Especially, when this information is not static and can vary considerably during time: new users can be added, others deleted, the privileges of some users can vary, etc.

3 Related Work

Numerous efforts have been made to add authorization support to Kerberos. In fact, the designers of the Kerberos protocol already anticipated this necessity and included an optional payload field for carrying authorization information. However, the format and specification of this field was left deliberately undefined.

The Sesame [7] protocol is aimed to solve the scalability problems of Kerberos caused by the utilization of symmetric key cryptography and the lack of an access control service. This protocol is based on adding to the Kerberos KDC a new service, known as PAS (Privilege Attribute Server). Thus, when a user issues a request to the KDC it is not only authenticated, but also provided with a PAC (Privilege Attribute Certificate) which can be presented to a server when necessary.

Sesame states that it can be implemented with both public key cryptography and symmetric cryptography, but this is not completely true. Even though the authentication of the user can be carried out making use of the standard Kerberos protocol, Sesame uses public key technology in different steps. For instance, the PACs are attribute certificates which are signed by the PAS, so the PAS must own a private/public key pair. Moreover, for the inter-domain operation the KDCs make use of public key technology. In this sense, it must be taken into account that asymmetric encryption techniques are considerably more demanding than the symmetric encryption techniques from the execution time and energy consumption point of view [8], [9], and in a ubiquitous environment the energy saving can be of vital importance.

In the IDfusion [10] protocol the authors try to combine the advantages of Kerberos and LDAP with the aim of creating a system that implements authorization functionalities. IDfusion is based on the utilization of a new type of identities, known as Service Instance intrinsic identities (SIii), which represent the willingness to convey authorization to a given user for a specific service. These identities are calculated as a combination of the identity of a user (Uii) with the identity of a service (Sii), and they include a cryptographic operation using the symmetric key of the protected service. The presence or absence of a given SIii object in the directory is translated as the willingness to provide or not the access represented by that SIii value. The current implementation of IDfusion makes use of the authorization payload field of a Kerberos Service Ticket to transport the SIii value.

Although its designers claim that IDfusion is a simple system, actually it is not a lightweight protocol, due to the fact that identities are represented as N-bit vectors and cryptographic hashes, and it makes extensive use of XML. On the other hand, as SIii identities are generated as a combination of Uii and Sii identity pairs, this system presents scalability problems. In an environment with N users that can access M services in order to perform P different operations over each one, NxMxP SIii identities will have to be created and stored. Furthermore, as SIii identities are generated from individual Uii identities, authorization management can not be based on groups of users. Additionally, IDfusion does not provide any mechanism for inter-domain operation, which penalizes even more the scalability of the system.

The researches in [11] propose an authorization and accounting system which focuses on the use of *restricted proxies*. A *proxy* is defined as a kind of token that allows a given entity to act with the rights and privileges of the principal that created it. A *restricted proxy* is defined as a proxy to whom certain conditions have been imposed. In this system the concept of an authorization server is introduced, but this server does not directly specify if a given principal has the right to access a specific service or object. Instead, when the authorization server receives a request from an authorized principal, it issues a restricted proxy that allows the authorized principal to act as the authorization server with the aim of asserting the principal's rights to access certain services or objects. In each case, the authorization server includes in the proxy the actions for which the principal is authorized in the form of restrictions.

This system can be implemented with public key cryptography as well as with Kerberos. When implemented with public key cryptography, thanks to the use of certificates, a single proxy can be used to grant access to multiple services or resources. When using Kerberos, however, a proxy is actually represented by a Kerberos Service Ticket and *authenticator*, and the restrictions are introduced in the ticket's authorization payload field. Therefore, as each Service Ticket is encrypted with the secret key of the principal for which it was issued, in this case it is necessary to generate a different proxy for each of the services that are wanted to be granted access. This derives in an increased complexity and scalability problems.

Microsoft introduced Kerberos as the default authentication protocol in Windows 2000 [12]. As in Sesame, Microsoft's solution for implementing authorization is based on the concept of Kerberos PAC (Privilege Attribute Certificate). A PAC contains the principal's group membership list which is required to create the *token* used by Windows clients to make access control decisions.

The PAC is sent along with the service request messages, and it is processed by the LSA (Local Security Authority) of the resource server, which validates it and generates the client's access token, for use in subsequent authorization decisions. In the access token the LSA includes the client's authorization data found in the Service Ticket's PAC and the client's authorization data found in the local security base. Therefore, the authorization data regarding a client is not centralized in a common place where it can be easily accessed and modified by a system administrator. Instead, it is distributed among the Active Directories and domain controllers, and every application server to which the user could eventually request access.

Finally, it must be noted that these approaches to add authorization support to Kerberos, just implement the necessary mechanisms to provide the end systems with the information required to take the authorization decision. However, in all these models are, in fact, the end servers which must perform the actual authorization of the requesting clients, either by querying local access control lists or by the enforcement of other mechanisms.

4 Proposed Solution

We believe that in a heterogeneous environment where users can be added or deleted frequently and the users' rights change regularly, it is preferable to maintain all the user-related information in a centralized location, where it can be easily updated by a system administrator. Moreover, we also believe, that a system containing all the information regarding user identities and user rights is the perfect location where authentication and authorization decisions should be performed. According to this, we have modified the operation of the standard Kerberos KDC so that apart from authenticating client identities, it also verifies their user rights, and only generates Kerberos Service Tickets for those users authorized to access the desired service.

4.1 The Time Synchronization Problem

Kerberos makes use of timestamps as a way of probing the freshness of the messages, and thus avoiding reply attacks. Therefore, the system clocks of all the entities that make use of the Kerberos services must be synchronized within the allowable clock-skew window. This necessity for synchronized clocks constitutes one of the major drawbacks of using timestamps. On the other hand, the use of timestamps allows Kerberos to be stateless; state is represented through the Kerberos tickets. Statelessness is extremely valuable from the scalability point of view, especially in an environment where the majority of the targeted applications are based on simple and stateless, request-response protocols.

Given the difficulty of maintaining synchronized clocks, we propose a *nonce*-based implementation of Kerberos. 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. This fact represents an important benefit for the operation of the whole system, since a given sensor or electrical appliance can be switched off without the hitch of loosing the state-related information. Basically, this implementation uses the *authtime* field of the Kerberos tickets and protocol messages to include a

nonce value which should be validated by each entity that receives such a ticket. In order to allow the validation of the received nonce values, we have introduced the concept of a *Nonce Validation Service* (NVS). The NVS is a new service that resides in the Kerberos KDC, together with the Authentication Server (AS) and the Ticket Granting Server (TGS).

The NVS maintains an information base in which each entry corresponds to a client and service principal pair and their associated nonce value. This information is updated by the TGS each time it generates a new Service Ticket with a given nonce value; and by the AS each time it generates a new Ticket Granting Ticket (TGT), which is in fact a service ticket to authenticate to the security principal representing the TGS. On the other hand, each entry has a limited lifetime, which coincides with the lifetime of the TGT or Service Ticket in which the nonce value associated to that entry was embedded. When the lifetime associated with an entry expires, the entry is deleted from the information base.

Additionally, this system allows the establishment of concurrent communications between the same principals: if multiple communications are established between the same client and service principals, there will be several entries regarding the given client/service pair in the information base maintained by the NVS.

Although we have obviated the timestamps included in the *authenticators* of the Kerberos protocol, it must be noted that these timestamps are also replaced by nonce values. In fact, they are replaced by the same nonce values embedded in the tickets the authenticators are sent with.

4.2 Authorization Information

We consider two types of security principals:

- Client principals, which represent the identities of the users that want to get access to services or resources.
- Service principals, which represent the identities of the end applications or resources that are wanted to be protected.

Regarding the service principals, it must be taken into account that many services or resources can be subject of different operations. This is the typical case of a given variable or parameter in a sensor, where the parameter can be subject of reading or writing operations. In such cases, we define the service principal identity as the operation to be performed over the final resource. Therefore, a single service will own as many service principals as operations can be performed over that service. That is, reading a variable would be considered a different service principal than writing it.

We propose an authorization solution which makes use of a role-based access control (RBAC) model [13], [14] in order to manage authorization-related information. This information is stored in an information base included in the Kerberos KDC. The information base contains two types of entries. On the one hand, it contains entries associating the client principals with a list of the roles that can be undertaken by them. There will be an entry of this type for each of the client principals defined in the Kerberos identity repository, and each client principal should be assigned at least one role. On the other hand, it contains entries associating the service principals with the different roles that should be granted access to the given service. As in the previous case, it will exist an entry of this type for each service principal defined in Kerberos;

and to each of them at least one role should be granted access, since it has no sense providing a service which no one will be able to access.

4.3 Obtaining the Service Ticket: The Authorization Decision

The authorization decision is performed by the KDC whenever a client principal requests a Service Ticket. At this point, the KDC owns all the necessary information to authenticate the requesting principal, and authorize or deny its access to the service for which it is requesting a Service Ticket. The Service Ticket request is authenticated, as in Kerberos, by the presentation of a previously obtained Ticket Granting Ticket (TGT) and the use of a session key. Once the requesting client principal has been determined to be a legitimate user, the KDC proceeds to verify if this principal owns the necessary rights to access the desired end service. With this purpose, the KDC queries its local authorization base, using as entry points the identities of the requesting client principal and the requested service principal. Then compares the roles with access right to the service with the roles that can be undertaken by the user, and if a match is found, the user is determined to be authorized to access the desired service.

In the case that a client principal is determined to be authorized to access the service principal for which it requested a Service Ticket, and only in this case, the requested Service Ticket is generated by the KDC. This way, the network load, as well as the data processing performed by the service principals, is reduced. Actually, the generation of a Service Ticket involves some resource consumption: first, 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; second, CPU consumption, mainly in the resource provider, as it has to decrypt the received Service Ticket and perform the authentication and authorization 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, in our model, a Service Ticket conveys both authentication, as well as authorization rights for the principal that owns it. In the generated Service Tickets the authorization payload field is used to embed the identifier of the role undertaken by the client principal when accessing a resource with the present ticket. If there is more than one match between the list of roles that can be assumed by the client principal and the list of roles with access to the requested service principal, the authorization payload field will contain the identifiers of all the roles that match the condition. This information is encrypted with the secret key of the service principal for which the ticket is intended. This way it is guaranteed that only this principal can read the authorization information embedded in the ticket, and it is also avoided the possibility of third parties generating their own authorization data and sending it to the KDC as encrypted authorization data to be included in the tickets.

4.4 The Service Access Phase

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 do not need to perform further authorization checks, or implement access control mechanisms as they 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, guaranteeing this way that it was generated by a trusted KDC. Second, the validity of the nonce value contained in the ticket must be checked against the Kerberos KDC. With this purpose a request must be sent to the NVS, as shown in Fig. 1.

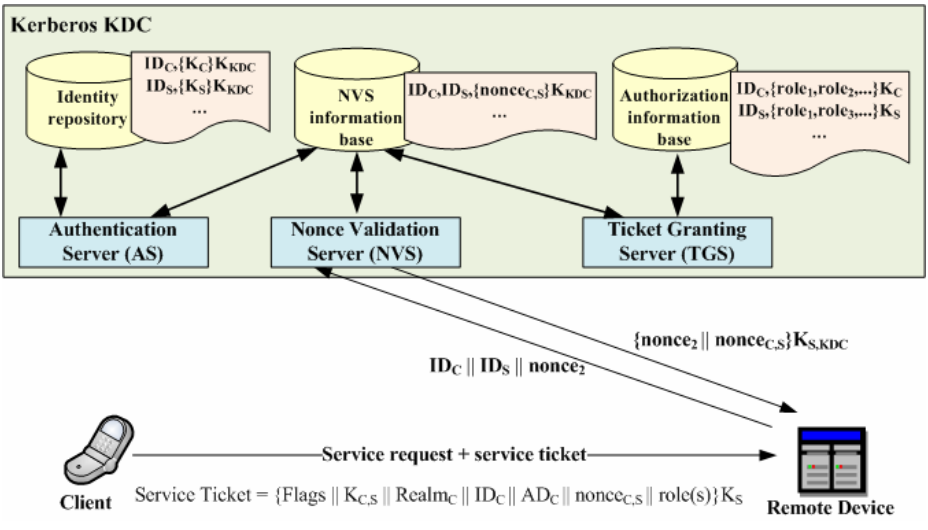

Fig. 1. Basic architecture and message exchanges of the proposed solution

In order to support the operation of the NVS, two new messages have been introduced into the standard Kerberos protocol:

- KRB_NVS_REQ: a short cleartext message specifying the identity of the client principal who sent the Service Ticket (ID_C) and the identity of the service principal that received it (ID_S). Additionally, it includes a nonce value in order to avoid reply attacks ($nonce_2$). This nonce value serves to associate the response from the NVS with the current request

$$ID_C \parallel ID_S \parallel nonce_2$$

- KRB_NVS_REP: this message just includes two nonce values, and it is encrypted with the secret key shared between the service principal and the KDC ($K_{S,KDC}$). This is not the service principal's secret key, but the session key provided by the KDC along with the TGT during the authentication process, as defined in the standard Kerberos protocol. However, in the standard Kerberos protocol, as the NVS does not exist, this key is shared only between the principal and the TGS. In the following line, the curly braces

denote encryption of the information contained between them with the secret key specified just after.

$$\{nonce_2 \parallel nonce_{C,S}\}K_{S,KDC}$$

The first nonce value is the same as the one received in the request ($nonce_2$) and serves to match the response with the corresponding request. The second nonce value ($nonce_{C,S}$) is the nonce value that should be embedded in the received Service Ticket. By comparing this value with the one extracted from the Service Ticket, the service principal can determine the validity of the received ticket.

As the information included in the KRB_NVS_REQ is not authenticated nor encrypted, the NVS does not know whether this information is actually true. However, it generates a reply and sends it to the requesting entity without knowing or caring if it is really who it claims to be, or an impostor. This behaviour is acceptable because the response is encrypted with a secret session key only known by the KDC and the service principal indicated in the request. As a result, nobody but this principal will be able to decrypt and use the reply message.

Nevertheless, an attacker could send numerous KRB_NVS_REQ messages in order to get known plaintext encrypted with the session key used by some principal to communicate with the KDC, and thus attack this key. Taking into account the short length of the reply message, the number of requests sent by a malicious entity in order to perform such an attack would have to be extremely high, and thus easily detectable by any Intrusion Prevention System (IPS). Anyway, even if a malicious entity reached to get the attacked key, what it would obtain is just a temporary session key with a limited lifetime, which will be overwritten with the next TGT renewal process. As a result, the consequences of the attack would be limited.

Someone may think that conveying authorization rights by the mere issuance of Kerberos Service Tickets is not suitable as there is no way of guaranteeing that the authorization process has been actually performed. For this reason, in our approach when the TGS issues a Service Ticket, it embeds in the authorization payload field the identification of the role the user is assuming when using that ticket. The existence of this information is the third of the main validity checks that are performed by the end applications over the received Service Tickets. Additionally, this information could also be used by the end entities to perform further tasks, such as accounting.

4.5 Revocation of User Rights

Thanks to the usage of nonce values as a method to guarantee the freshness of the Kerberos tickets, the revocation of users' rights can be performed easily, just by deleting from the NVS information base the entries associated with the user and the service for which access is wanted to be denied.

When a system administrator modifies the list of roles associated with a given client principal or service principal identity, automatically all entries associated with the modified principal are deleted. This way, if a client tries to access a resource server by presenting an old Service Ticket, the nonce value included in the ticket and the one provided by the NVS during the ticket validation process will not match up, and thus,

the service will not be provided. In this case, the client will have to request a new Service Ticket to the Kerberos TGT, which will perform the validation of the client's access rights, using this time the newly updated values.

However, when the modification of the authorization information occurs, the already established connections are not cut off, and they continue until the Service Ticket expires and a new one has to be requested, in which case the authorization process is performed again.

4.6 Inter-domain Operation

The development of networking technologies has made the collaboration between geographically separated enterprises and organizations easier. Thanks to these technologies organizations can easily exchange information and work together. However, for a successful data sharing and cooperative work, it is essential to implement secure and reliable communication mechanisms. User authentication and access control are two important issues in this regard.

The Kerberos protocol allows the existence of different realms or domains, each one with its own KDC. The inter-domain operation of Kerberos relies on the concepts of *referral tickets* and *inter-domain secrets*. To enable inter-domain authentication, every domain that is trusted by another domain is registered in the domain's KDC as a security principal. These principals' secret keys are usually known as inter-domain secrets. When a client principal in a given domain wants to obtain a Service Ticket for a service principal in a different domain, it has to query the remote domain's KDC. Before the client principal can contact the remote domain's KDC, it has to get from its own KDC a valid Service Ticket to talk to the remote KDC. This Service Ticket, which is encrypted with the inter-domain secret, is known as a referral ticket. It must be noted that the trust relationship between two domains may not always be direct; it can also be based on a tree hierarchy. If the trust relationship is established as a hierarchy, each domain is registered in its parent domain's KDC, and trust paths have to be followed in order to obtain a referral ticket for a domain in a different branch of the tree.

Regarding our approach, referral tickets convey, embedded in their authorization payload field, information indentifying the different roles the user is allowed to undertake in its local domain. When the remote domain's KDC receives a Service Ticket request authenticated with a referral ticket, it extracts from this ticket the list of roles the requesting principal can execute and compares them with the roles allowed to access the desired service. If a matching entry is found, the remote KDC generates a Service Ticket, including in the authorization payload field an identifier of the role the user is assuming; if not, no Service Ticket is generated, and an authorization error is returned to the requesting client principal. In order to facilitate the inter-domain operation between organizations with a collaborative relationship, some defined roles can extend over different domains. That is, the same role may be used to grant access to different services in different domains.

Thanks to these mechanisms, the Service Ticket issued by the remote domain's KDC for the remote client principal is identical to the one it would have issued for a

local client principal. Therefore, when a service principal receives a request from a client of a different domain, it does not even realize that the request came from outside its domain, and it processes it exactly like any other incoming request.

5 Conclusions

Authentication and authorization are very closely related concepts. In fact, the final aim of these two processes is to guarantee that only the legitimate end users allowed to access a certain service or resource are the ones who actually access it. For this reason, we think that it is not worth providing a user with authentication credentials for a service it is not authorized to access. Therefore, we have developed a modification of the Kerberos protocol, so that before providing a user with the necessary credentials to authenticate to a service, it verifies if the user is allowed to access this service, and only in this case, generates the requested credentials. This way, users are only able to authenticate to services they are authorized to access. This supposes a resource saving for the end services, as they do not need to process authentication requests of users which in the end will not be able to access the service, as they are not authorized to do so.

We think that the best place to enforce the access control restrictions is the Kerberos authentication server. Kerberos provides the users with the necessary credentials to access a service in two steps: first, it authenticates the identity of the users and provides them with the necessary elements to establish secure and authenticated communications with the KDC. Then, it generates specific credentials for each service the users want to communicate with. It is before the second step takes place where the authorization rules should be enforced, limiting this way the generation of authentication credentials to legitimate authorized users. As a consequence, the generated credentials do not only guarantee the identity of the user they represent, but also that this user is authorized to access the service for which the credentials are intended. Thanks to this, the end application servers do not need to implement and enforce their own access control restriction, as the Service Tickets generated by the KDC already convey authentication and authorization rights for the users that own them.

On the other hand, performing authentication and authorization of users in the same server facilitates the user management processes. As all the user-related information is centralized in a single server, the network administrator only needs to interact with one server in order to add or delete users, or modify the users' access rights.

Finally, it should also be mentioned that due to the fact that it provides Single Sign-On functionalities, Kerberos is a very suitable protocol to be used in organizational environments. The user only needs to enter his or her password in the first step of the authentication process, where in fact, his or her identity is verified. From then on, the user owns all the necessary information to authenticate to the Kerberos KDC and obtain the Service Tickets needed to access all the services it is allowed to, without needing to enter the password again.

References

1. Al-Muhtadi, J., Mickunas, D., Campbell, R.: Wearable security services. In: 21st International Conference on Distributed Computing Systems, Phoenix, pp. 266–271 (2001)
2. Abadi, D.J., Lindner, W., Madden, S., Schuler, J.: An integration framework for sensor networks and data stream management systems. In: 30th international Conference on Very Large Data Bases, Toronto, vol. 30, pp. 1361–1364 (2004)
3. Kansal, A., Goraczko, M., Zhao, F.: Building a sensor network of mobile phones. In: 6th international conference on Information processing in sensor networks, Cambridge, Massachusetts, pp. 547–548 (2007)
4. Dunkels, A., Alonso, J., Voigt, T.: Making TCP/IP Viable for Wireless Sensor Networks. In: Work-in-Progress Session of the first European Workshop on Wireless Sensor Networks, Berlin (2004)
5. 6lowpan IETF group, `http://www.ietf.org/html.charters/6lowpan-charter.html`
6. Neuman, C., Hartman, S., Raeburn, K.: The Kerberos network authentication service, v5 (2005), `http://www.ietf.org/rfc/rfc4120.txt`
7. Kaijser, P., Parker, T., Pinkas, D.: SESAME: the solution to security for open distributed systems. Computer Communications 17(7), 501–518 (1994)
8. Ruangchaijatupon, N., Krishnamurthy, P.: Encryption and power consumption in wireless LANs-N. In: 3rd IEEE Workshop on Wireless LANs, Newton, Massachusetts (2001)
9. Potlapally, N.R., Ravi, S., Raghunathan, A., Jha, N.K.: Analyzing the energy consumption of security protocols. In: 2003 International Symposium on Low Power Electronics and Design, Seoul, pp. 30–35 (2003)
10. Wettstein, G.H., Grosen, J.: IDfusion, an open-architecture for Kerberos based authorization. In: AFS and Kerberos Best Practices Workshop, Michigan (2006)
11. Neuman, C.: Proxy-based authorization and accounting for distributed systems. In: 13th International Conference on Distributed Computing Systems, Pittsburgh, pp. 283–291 (1993)
12. Walla, M.: Kerberos explained, issue of Windows 2000 Advantage magazine (2000), `http://technet.microsoft.com/en-us/library/bb742516.aspx`
13. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer 29(2), 38–47 (1996)
14. Ferraiolo, D.F., Sandhu, R.S., Gavrila, S., Kuhn, D.R., Chandramouli, R.: Proposed NIST standard for role-based access control. ACM Transactions on Information and System Security 4(3), 224–274 (2001)