Computer Standards & Interfaces 33 (2011) 494–504

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Providing EAP-based Kerberos pre-authentication and advanced authorization for network federations

Rafael Marín-López *, Fernando Pereñíguez, Gabriel López, Alejandro Pérez-Méndez

Dept. Information and Communications Engineering (DIIC), University of Murcia, 30100, Spain

ARTICLE INFO

Article history:
Received 25 June 2010
Accepted 22 February 2011
Available online 31 March 2011

Keywords:
AAA
Authentication
Authorization
EAP
Kerberos
SAML
XACML

ABSTRACT

Kerberos is a well-known standard protocol which is becoming one of the most widely deployed for authentication and key distribution in application services. However, whereas service providers use the protocol to control their own subscribers, they do not widely deploy Kerberos infrastructures to handle subscribers coming from foreign domains, as happens in network federations. Instead, the deployment of *Authentication*, *Authorization* and *Accounting* (AAA) infrastructures has been preferred for that operation. Thus, the lack of a correct integration between these infrastructures and Kerberos limits the service access only to service provider's subscribers. To avoid this limitation, we design an architecture which integrates a Kerberos pre-authentication mechanism, based on the use of the *Extensible Authentication Protocol* (EAP), and advanced authorization, based on the standards SAML and XACML, to link the end user authentication and authorization performed through an AAA infrastructure with the delivery of Kerberos tickets in the service provider's domain. We detail the interfaces, protocols, operation and extensions required for our solution. Moreover, we discuss important aspects such as the implications on existing standards.

1. Introduction

The impressive growth of telecommunications has promoted the establishment of business agreements between service providers (SPs) within the so-called *federations*, in order to increase the revenues of the deployed network services. Indeed, such network federations allow a service provider's subscriber to access the services of affiliated providers in the federation.

In general, a subscriber can access network services within a federation by means of a single authentication process, which is commonly named *single sign-on* (SSO) access. Nowadays, Kerberos is becoming one of the most widely deployed standards for authentication and key distribution providing this feature [1]. Indeed, operating systems and different network applications (FTP, SSH, HTTP, etc.) are already integrating Kerberos to perform service access control. However, whereas service providers use this protocol to control the access of their own subscribers, they do not usually deploy Kerberos multi-domain (*cross-realm*) infrastructures to control subscribers coming from a different domain in the federation. Therefore, the only way for these subscribers can access the services is to have a pre-established security association with the service provider's local (*single-realm*) Kerberos infrastructure, which creates a serious deployment issue. Moreover, the authorization management has not been integrated in the protocol.

Instead, service providers typically deploy *Authentication*, *Authorization* and *Accounting* (AAA) infrastructures for controlling these operations in federated networks. Furthermore, the *Extensible Authentication Protocol* (EAP) [2] has become one of the most promising candidates to provide flexible authentication and easy integration with underlying AAA infrastructures. For example, *eduroam* [3] deploys an EAP/AAA-based roaming infrastructure used by the international research and education community of more than five hundred institutions.

Thus, the lack of a correct integration between AAA infrastructures and Kerberos may seriously limit the service access only to subscribers (hereafter *end users*) that belong to the service provider's domain, by disallowing one of the most important goals in network federations: to allow any end user in the federation to access any application service no matter the domain. This has motivated an incipient effort in standardization bodies [4] to integrate AAA infrastructures for service access control in network federations.

In this sense, the main goal of this work is to solve these problems by defining an unified architecture that allows to integrate existing AAA infrastructures in the federation with the service access control based on the use of Kerberos in the service provider's domain. Specifically, we have designed a novel Kerberos pre-authentication mechanism [5] based on EAP, which allows end users of any domain in the federation to authenticate themselves in the service provider's domain with their home domain credentials, by leveraging the deployed AAA infrastructure that interconnects the service provider's

* Corresponding author at: University of Murcia, Facultad de Informatica, Campus de Espinardo S/N, 30100, University of Murcia, Spain. Tel.: +34 868 88 85 01; fax: +34 868 88 41 51.
E-mail addresses: rafa@um.es (R. Marín-López), pereniguez@um.es (F. Pereñíguez), gabilm@um.es (G. López), alex@um.es (A. Pérez-Méndez).

0920-5489/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.csi.2011.02.005

domain and the end user's home domain. As a consequence, our proposal avoids the need of deploying a Kerberos cross-realm infrastructure. Moreover, in order to provide advanced authorization management to our architecture, we have integrated the well-known standards SAML [13] and XACML [17].

The rest of this paper is structured as follows. Section **2** provides an overview of the main technologies used in this proposal. Section 3 presents the proposed architecture. Section 4 details the end user authentication phase, where we propose two alternatives. In Section 5 we describe how advanced authorization management could be integrated in this architecture and Section 6 discusses some aspects that must be taken into account in the deployment of our solution. Section 7 describes the related work and, finally, Section 8 presents some conclusions and future work.

2. Background

In the following, we provide some basic concepts about the different protocols and technologies that integrate our solution.

2.1. Kerberos

Kerberos [5] is a secure three-party authentication and key management protocol based on symmetric keys. Three entities are involved: a *client*, an *application server* providing a service, and a *key distribution center* (KDC). The KDC is composed of two special servers: an *Authentication Server* (AS) and a *Ticket Granting Server* (TGS). The following trust relationships are required to be pre-established: AS ⇔ TGS, client ⇔ AS and application server ⇔ TGS.

Fig. 1 shows a typical Kerberos exchange. Initially, the client requests a *Ticket Granting Ticket* (TGT) from the AS through a *KRB_AS_REQ/KRB_AS_REP* exchange (1). The TGT is a special ticket used for generating other tickets. The AS generates a *TGS session key* that is included in the TGT and sent to the client in the *KRB_AS_REP* message. Kerberos implements a mechanism called *pre-authentication* that allows the KDC to authenticate the client before providing the TGT. When using multi-roundtrip authentication mechanisms, client and AS can exchange authentication information through several *KRB_AS_REQ/KRB_ERROR* exchanges. When the authentication process succeeds, the AS responds with a final *KRB_AS_REP* containing the requested TGT.

Once the client owns the TGT, it can request a *Session Ticket* (ST) to the TGS for accessing a specific service (2). With this purpose, the client sends a *KRB_TGS_REQ* protected with a checksum computed with the TGS session key. When the TGS validates the TGT, it generates a *session key* that is included in both the ST and the *KRB_TGS_REP* message. Finally, similarly to the TGS exchange, the client authenticates itself to the service (3) by sending the ST to the application server in a *KRB_AP_REQ* message. Optionally, a *KRB_AP_REP* message can be used if the client needs to authenticate the application server.

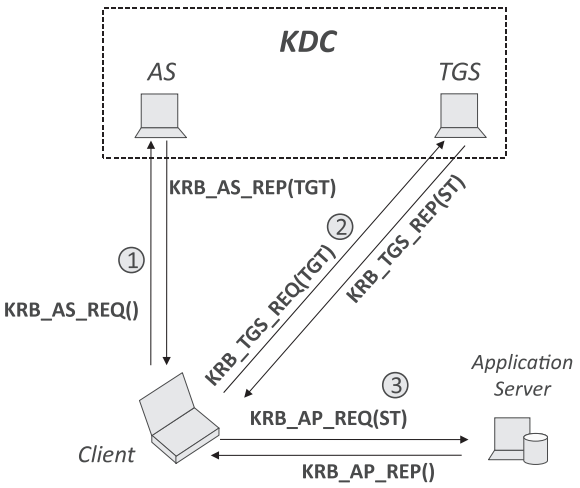

Fig. 1. Kerberos standard signaling.

Kerberos supports an additional operation mode called *cross-realm authentication*. This variant allows a client to authenticate itself against services located in foreign realms. Thanks to trust relationships pre-established between TGS/KDCs from different realms, the client follows the path from the home to the visited TGS by acquiring the so-called *cross-realm* TGTs. Finally, the client contacts the visited TGS/KDC and obtains the ST that will be presented to the service. Note that we will use the term *domain* instead of *realm* hereafter.

2.2. Generic Security Service Application Program Interface

The *Generic Security Service Application Program Interface* (GSS-API) [6] is a generic framework that provides security services like authentication, integrity and confidentiality. Distributed network applications that need to protect their communications can employ the different security services offered by the GSS-API and remain independent from a particular security mechanism. The most extended GSS-API implementation is provided by the Kerberos protocol [7], though other mechanisms such as EAP have been also proposed [8].

The GSS-API allows a peer (*GSS-API Initiator*) to establish a *security context* with another peer (*GSS-API Acceptor*). The negotiation of the context starts when the initiator invokes the *GSS_Init_sec_context()* function, which indicates a pending status (GSS_S_CONTINUE_NEEDED) to complete the context establishment, and returns a token to be passed to the acceptor.

The acceptor passes the received token to the *GSS_Accept_sec_context()* function. Assuming a single round-trip authentication protocol, the function indicates a GSS_S_COMPLETE status and returns a token to be sent back to the initiator. Finally, the initiator invokes again the *GSS_Init_sec_context()* function (passing the received token) which returns the status of the context establishment.

Once the security context has been established, the application (either the initiator or acceptor) can perform the message protection. The GSS-API offers two different mechanisms to protect application messages. On the one hand, through the *GSS_GetMIC/GSS_VerifyMIC* calls, messages are authenticated and integrity protected. On the other hand, by using the *GSS_Wrap/GSS_Unwrap* functions, confidentiality is also provided.

2.3. The Extensible Authentication Protocol

The *Extensible Authentication Protocol* (EAP) [9] is a request/response protocol which allows different types of authentication mechanisms named *EAP methods* (e.g., based on symmetric keys or certificates). The EAP methods are run between an *EAP peer* and an *EAP server* through an *EAP authenticator*. From a security standpoint, the authenticator does not take part in the mutual authentication process but acts as a packet forwarder.

To carry out an EAP authentication, the authenticator usually starts the process by requesting the peer's identity through an *EAP Request/Identity* message. The peer answers with an *EAP Response/Identity* containing its identity. With this information, the server will select the authentication method to be performed. The method execution involves several exchanges *EAP Request/Response* between the peer and server.

The most typical configuration is the so-called *pass-through* authenticator model (Fig. 2) where the EAP server is placed on a AAA server and the EAP authenticator is implemented in a separated